Design System além do layout

Criação, estratégia e governança

Casa do Código | alura

MATEUS VILLAIN

Sumário

- ISBN
- O autor
- Agradecimentos
- Introdução
- Parte 1 - Entendendo o design system
  - 1 Um produto chamado "design system"
  - 2 Solucionando problemas
- Parte 2 - Planejamentos iniciais
  - 3 Primeiras definições
  - 4 Equipe
  - 5 Auditoria de UI
  - 6 Escolhendo peças, produtos e pessoas
  - 7 Roadmap
- Parte 3 - Governança
  - 8 Consistência da marca
  - 9 Responsabilidades da equipe
  - 10 Política de versionamento
  - 11 Princípios de design
- Parte 4 - Estilos e design tokens
  - 12 Foundation
  - 13 Acessibilidade
  - 14 Cores
  - 15 Elevação
  - 16 Espaçamento
  - 17 Tipografia
  - 18 Design tokens
- Parte 5 - Componentes e microinterações
  - 19 Componentes
  - 20 Propriedades e customizações
  - 21 Estados
- Parte 6 - Boas práticas de documentação
  - 22 Iniciando a documentação

    - 23 Documentando componentes
    - 24 Documentando os patterns
- Parte 7 - Conclusão
    - 25 Glossário
    - 26 Referências

ISBN

Impresso: 978-85-5519-408-5

Digital: 978-85-5519-407-8

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Villain, Mateus
Design system além do layout : criação, estratégia e governança / Mateus Villain. -- 1. ed. -- São Paulo : AOVS Sistemas de Informática, 2025.

ISBN 978-85-5519-408-5

1. Design 2. Design - Desenvolvimento 3. Design - Estudo e ensino 4. Design (Projetos) 5. Tecnologia I. Título.

25-306732.0 CDD-660.02

Índices para catálogo sistemático:

1. Tecnologia e design 660.02

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

O autor

Mateus Villain é um designer criciumense que atua com produtos digitais desde 2016. Ao longo da carreira, trabalhou em diferentes empresas de tecnologia, iniciou sua carreira como professor em 2021 ministrando cursos sobre UX, UI e design system na Alura, e hoje é Product Designer na Locaweb, onde integra o time responsável pelo design system da empresa.

É graduado em Design pela UniSatc e está cursando pós-graduação em Digital Product Management pela FIAP, com foco em se especializar na liderança e no gerenciamento estratégico de produtos digitais.

Agradecimentos

Quero começar agradecendo ao Luiz Lima, que me deu o empurrão inicial para começar a escrever este livro, e ao Bruno e Will que contribuíram com feedbacks e revisões.

Impossível não agradecer também a minha esposa e a minha família pelo apoio e por ficarem tão animados quanto eu com esse projeto.

E para fechar, um muitíssimo obrigado a Casa do Código, em especial a Vivian Matsui, por abrir as portas para mim, e por toda a paciência durante o período de escrita.

Introdução

Eu lembro que, quando ouvi falar pela primeira vez de **design system**, foi de uma forma totalmente aleatória em um artigo na internet sobre padronização de layouts para interfaces digitais. Minha memória não é muito clara sobre o contexto do assunto que me levou a ter interesse em pesquisar sobre isso, mas o que eu não consigo esquecer é que, naquele momento, senti uma curiosidade enorme, que logo se transformou em entusiasmo quando eu descobri a existência desse conceito.

Se eu parar para pensar hoje, era bem óbvio que seria algo criado uma hora ou outra, porque existem produtos longos, com interfaces complexas, e, com certeza, seria necessário criar um meio de padronizar tudo isso, já que facilitaria muito o trabalho das pessoas que criam esses produtos. Mas, para a minha surpresa, conforme fui estudando, notei que se tratava muito mais do que padronização visual — que era só um pedaço do enorme quebra-cabeça que é o design system. E a minha diversão só estava começando.

Atualmente, no Brasil, essa área tem se popularizado muito, e diversas empresas que atuam com tecnologia passaram a perceber que isso não era um simples bônus, mas uma necessidade real. À medida que o negócio cresce, mais pessoas começam a se interessar e a consumir os produtos ou serviços da marca, que, por sua vez, precisa cumprir sua missão de manter e melhorar a experiência dessas pessoas naquilo que elas se propuseram a fazer. E, mesmo que o design system não seja o produto com o qual essas pessoas usuárias vão interagir diretamente, ele terá o papel fundamental de fidelizar cada vez mais o público com a marca e será responsável por proporcionar uma experiência de usabilidade única e memorável.

Pense no que seria de uma *big tech* como o Google — empresa que possui uma variedade enorme de produtos e é responsável tanto pelo maior buscador do mundo quanto pelo sistema operacional

mais usado em dispositivos móveis — se não tivesse o seu design system: o *Material Design*. Uma das grandes vantagens de utilizar os serviços do Google, como o Gmail, o Drive, o Calendar ou o Photos, está nas similaridades e facilidades de uso entre eles. Afinal, depois de aprender a usar um desses produtos, utilizar qualquer outro se torna muito mais fácil — mesmo que tenham funções diferentes. Existe uma identidade clara nos posicionamentos, interações, nomenclaturas, elementos visuais e até mesmo na acessibilidade.

Por ser algo relativamente novo, principalmente no nosso país, é necessário promover a conscientização para que todas as pessoas que atuam com tecnologia compreendam que um design system não apenas facilita e agiliza o desenvolvimento de um produto, como também contribui para a fidelização do público.

Além disso, é preciso ter ciência de que um design system, apesar de ter *design* no nome, não é feito nem mantido somente por designers, mas sim por uma equipe multidisciplinar, que dedica atenção contínua à sua evolução. Por essa razão, este livro não se destina apenas a designers. Ele também não é um guia prático sobre como criar seu design system em ferramentas de prototipagem, nem indica quais linguagens de programação devem ser utilizadas para implementá-lo.

A proposta aqui é apresentar os conceitos essenciais para quem atua com design system — conhecimentos que vão muito além do ferramental. Afinal, quando lidamos com fundamentos, desde a definição de uma paleta de cores até a elaboração da documentação de uso, não faz sentido limitar o conteúdo a plataformas específicas.

Por fim, a motivação principal em publicar este livro é para que exista, de fato, uma interface-guia que ensine, para todos os níveis profissionais, com teorias e práticas, sobre a construção e evolução de um design system do início ao fim. Já vi surgirem muitos cursos e e-books brasileiros relacionados a esse tema, mas a área ainda carece de abordagens aprofundadas, que abordem os conteúdos reais, as bases de criação, referências e bons casos de uso.

Por isso, espero que, com este livro, qualquer pessoa que tenha o mínimo interesse na área de design system possa aprender mais, se desenvolver e — o mais importante de tudo — evoluir como profissional e levar todo o conhecimento para as práticas diárias.

Para quem é este livro

Como já vimos, este livro não foi feito somente para designers. Ao longo da leitura, você vai perceber que a linguagem é pensada para qualquer pessoa que tenha o mínimo interesse em saber como um design system é levantado e mantido — seja você designer, programador(a), gestor(a) que atua em um time de design system ou alguém que apenas quer conhecer mais sobre o assunto.

Quero ressaltar ainda que este livro foi pensado para profissionais de todos os níveis de experiência. Seja você uma pessoa em início de carreira (júnior), com alguma experiência (pleno), com bastante bagagem (sênior) ou até mesmo em uma posição de liderança, cada capítulo foi escrito com o cuidado de trazer detalhes que realmente contribuam para o seu aprendizado. O objetivo é preparar você para os desafios práticos do dia a dia de quem trabalha com design system.

Estrutura de estudo

É sempre bom frisar que um design system não nasce da criação direta de interfaces no Figma, nem mesmo por linhas de código. Embora a visão de fora possa parecer simples, há uma série de definições importantes que precisam ser tomadas antes da construção propriamente dita.

Não basta querer ter um design system. Antes mesmo de iniciar o processo, é crucial passar por determinados processos, para garantir que qualquer tomada de decisão seja assertiva e não ocorra a necessidade de voltar atrás.

Para isso, organizei este livro em seis partes, passando não só pelo conceito de design system, mas também pelas etapas de construção e evolução.

Parte 1 - Entendendo o design system

Na primeira parte, o foco será o entendimento definitivo do que é um design system. Das seis partes do livro, esta é a menor, mas ainda assim essencial para compreender o restante do conteúdo. Ela garante que você saiba o que é um design system e quais problemas ele se propõe a resolver.

Parte 2 - Planejamentos iniciais

Como eu já mencionei, a construção de um design system não deve iniciar pelo Figma. Nesta segunda parte, você entenderá melhor quais planejamentos e processos você e sua empresa devem tomar para "organizar a casa" corretamente e começar com as definições corretas.

Parte 3 - Processos de governança

Quem vê de fora pode pensar que um design system vive e respira apenas de aspectos visuais, mas esse produto só funciona de fato com a implementação dos processos de governança. A terceira parte abordará alguns desses processos, como princípios de design e responsabilidades da equipe, que garantem transparência, responsabilidade e eficiência na tomada de decisões.

Parte 4 - Estilos e design tokens

Todo design system precisa de informações que caracterizam o aspecto visual dos elementos, como cores e tipografia, de forma que sejam totalmente escaláveis. A quarta parte foca na acessibilidade

do seu produto, nos principais estilos trabalhados e em como podemos alavancar isso com os design tokens.

Parte 5 - Componentes e microinterações

Componentes são blocos reutilizáveis de interface, como botões ou campos de texto, que encapsulam funcionalidade e estilo, facilitando a construção e manutenção de interfaces de usuário consistentes. Nessa parte, você entenderá a definição completa, as classificações e como trabalhar com as microinterações dos componentes.

Parte 6 - Boas práticas de documentação

Finalizamos o livro com a sexta parte, encerrando o ciclo inicial com a documentação do seu design system. Você entenderá qual a importância da documentação para os colaboradores, quais informações são indispensáveis e as melhores práticas para sua produção.

Parte 7 - Conclusão

Aqui deixo referências para você aprofundar suas pesquisas, recomendações de materiais e eventos, a lista das ferramentas mencionadas no livro, bem como a bibliografia consultada.

Parte 1 - Entendendo o design system

Um design system é um conjunto de diretrizes e padrões que serve como base para o desenvolvimento de produtos digitais eficientes e consistentes. Ele vai muito além de meros elementos visuais, envolvendo princípios que orientam a interação e a usabilidade. Com uma abordagem bem definida, esse sistema não apenas garante o alinhamento de todos os aspectos da interface com a identidade da marca, como também facilita a colaboração entre diferentes equipes, permitindo que a organização responda de forma ágil às mudanças nas necessidades do mercado e nas expectativas das pessoas usuárias.

Nesta primeira parte do livro, você lerá sobre:

- A história do design system e sua evolução ao longo do tempo.
- O início das padronizações na web e seu impacto no design.
- Os principais problemas que um design system ajuda a resolver.

Capítulo 1
Um produto chamado "design system"

Um equívoco muito comum é achar que um design system seja apenas um projeto paralelo ou temporário dentro de uma empresa e que, ao ser “finalizado”, o trabalho esteja concluído. Já adianto: esse é um pensamento profundamente equivocado. A criação e manutenção de um design system envolvem um alto nível de complexidade, exigindo que cada decisão seja tomada com atenção e cuidado. Ao longo dos capítulos, você compreenderá cada detalhe com profundidade. Mas, antes de avançar, é essencial conhecer dois pontos fundamentais para que todo o conteúdo deste livro faça sentido.

O primeiro ponto a saber é: **design system é um produto que é consumido por outros produtos**. Pessoas não navegam por um design system, mas interagem diretamente com seus elementos por meio de sites, aplicativos e outros produtos digitais que consomem sua enorme biblioteca de padrões, estilos e componentes.

Por exemplo: se uma empresa possui três produtos diferentes, a adoção de um design system garante uma consistência visual forte e harmônica entre eles. É o que acontece com os produtos do Google — como Gmail, Calendar, Drive e muitos outros — que, apesar de terem propostas distintas, compartilham características visuais e de interação que os tornam facilmente reconhecíveis como parte do mesmo ecossistema.

O segundo ponto é: **não se trata de algo temporário**. Um design system só acaba quando um novo surge, ou quando uma empresa encerra suas atividades. Do contrário, ele precisa evoluir com o tempo, pois a tecnologia avança e melhora, tendências de design e comportamentos vão mudando e, se o seu design system não for atualizado, é provável que sua empresa pare no tempo.

Já pensou se até hoje, o padrão de estilo do iPhone ainda se baseasse no *skeumorfismo*, ou seja, em representar os elementos digitais de forma que se pareçam muito com objetos reais? Não se trata apenas de aprimorar a parte visual, mas de uma busca contínua por compreender e atender plenamente às necessidades das pessoas usuárias.

Trata-se de um produto tão grande e complexo que, para quem está começando a estudá-lo, é comum que surjam dúvidas — por exemplo, sobre a diferença entre um design system e um **guia de estilo**. É provável que você já tenha ouvido falar desses dois termos em contextos simultâneos, mas distintos.

A resposta para isso é bastante simples: o guia de estilo é só uma das partes que compõem um design system. Ele reúne os estilos visuais — como cores, tipografia, sombras, ícones e componentes —

e tudo o que diz respeito à hierarquia visual do produto. Esse conjunto se complementa com outros recursos, como guias de acessibilidade, interação, documentação e muitos outros fatores.

É por essa razão que ouvimos falar de guias de estilo sem design system, que normalmente são feitos para projetos menores, e que não há a necessidade de evoluir isso para algo mais complexo; porém, não é possível ter um design system sem um guia de estilo.

Podemos comparar um design system a uma biblioteca: nela, há livros e guias elaborados de maneira detalhada e minuciosa para garantir a compreensão no momento de construir o que está sendo proposto — que, nesse caso, são os produtos digitais.

Aqui estamos nós, falando sobre o que é um design system, com base no que observamos em grandes empresas do mercado como Google, Apple e IBM. Mas quantas vezes você já ouviu falar sobre a história de como tudo isso começou até chegarmos ao cenário atual? A prática é, sem dúvida, indispensável para o nosso trabalho, mas uma excelente forma de começar — ou de continuar se desenvolvendo — é conhecendo um pouco dessa trajetória. Como alguém que ama estudar e trabalhar com o tema, não poderia deixar de compartilhar um pouco dessa história com você.

1.1 O início da história

Se hoje estamos conversando sobre design system, talvez seja graças a uma pessoa específica que, há muitos anos — antes mesmo de a própria Web ter sido cogitada —, realizou uma pesquisa relacionada à padronização de produtos digitais: Christopher Alexander, austríaco, arquiteto, designer e autor de diversas publicações, entre elas a célebre *A Pattern Language*. Vamos por partes para entender a relevância dele nessa história.

Christopher Alexander nasceu em Viena, no dia 4 de outubro de 1936, e dedicou sua vida atuando principalmente como arquiteto e teorista de design. Nesse último campo, desenvolveu diversas hipóteses e conduziu pesquisas em áreas variadas, como urbanismo e design de software.

Bacharel em Arquitetura e Mestre em Matemática pela Universidade de Cambridge, durante sua carreira, seu foco foi a área de arquitetura, atuando na construção de mais de 200 edifícios em cinco continentes diferentes, além de ter sido professor na Universidade de Berkeley. Apesar disso, suas observações e pesquisas em diferentes contextos o levaram a explorar outras áreas, tornando-se conhecido como o pai do movimento Pattern Language, ou "Linguagem de Padrões". Em 1977, publicou um livro com o mesmo nome, que se tornou um grande sucesso de vendas.

O livro, escrito em colaboração com seus alunos, buscou validar e explicar uma padronização de linguagem que permitisse a qualquer pessoa construir produtos de forma mais prática. Com mais de 1.000 páginas, descreve 253 padrões diferentes e inspirou diversas pessoas e áreas, incluindo Will Wright, criador dos famosos jogos The Sims e SimCity.

É claro que, nessa época, antes mesmo da invenção da web, o modelo que Alexander havia elaborado não possuía tantos cenários e níveis de complexidade como temos hoje, com uma variação muito maior de dispositivos e produtos nos quais podemos atuar. Ainda assim, o sentido de construção e aprimoramento de um produto permanece o mesmo: prever os problemas mais comuns que podem acontecer nos mais variados ambientes que estamos envolvidos e, após testes e validações, projetar formas de contorná-los, criando um produto com identidade e harmonia.

Anos depois, surgiu um livro extremamente semelhante ao de Alexander, mas desta vez com foco específico em design de interfaces. Essa obra renovou e deu continuidade à temática da padronização visual, tornando-se um guia altamente recomendado

para qualquer pessoa que viesse a trabalhar diretamente com User Interface. O livro é de Jennifer Tidwell, designer de interfaces, consultora em diversas áreas do design de produtos digitais e autora do best-seller *Designing Interfaces*, que apresenta diversos padrões para aplicativos de smartphones, computadores, softwares e websites, oferecendo orientação útil para profissionais de todos os níveis.

Não apenas os livros possuem similaridade no conteúdo, como também a história de suas concepções. Tidwell começou a pensar sobre padronizações para interfaces digitais no final da década de 1990, enquanto estudava no MIT. Sua motivação principal para esse estudo era como uma linguagem padrão poderia ajudar designers no mundo todo na criação de produtos digitais, para que não precisassem ficar receosos ou pensativos a respeito de suas criações.

Durante alguns anos, ela dedicou grande parte de seu trabalho a realizar testes em diversos cenários já existentes, buscando maneiras de aprimorar o desempenho e a experiência do usuário. É claro que comportamentos mudam e tecnologias evoluem, mas os padrões inicialmente estabelecidos serviam a múltiplos casos. Com o surgimento de novos dispositivos e demandas, foram lançadas mais duas edições ao longo do tempo, incluindo informações atualizadas — até mesmo voltadas para social media — e novos exemplos.

A partir de 2005, começaram a surgir as primeiras bibliotecas de padrões visuais, inicialmente voltadas para sites, levando em conta que a Web 2.0 havia surgido um ano antes e já estava sendo amplamente explorada.

1.2 O início das padronizações na web

Há não muito tempo, em 1989, um britânico estava prestes a entrar para a história ao inventar algo que, hoje, permite que pessoas em todo o mundo se conectem, publiquem documentos e compartilhem conhecimento umas com as outras.

Sir Timothy John Berners-Lee, ou simplesmente Tim Berners-Lee, é um cientista da computação famoso por ter criado a World Wide Web, ou Web. É graças a ele que hoje podemos pesquisar informações no Google, postar memes na internet e acessar sites para comprar livros incríveis — como este que você está lendo agora.

No ano seguinte à criação da Web, Berners-Lee desenvolveu o primeiro navegador para acessar e compartilhar informações. Porém, só em agosto de 1991 que todo esse novo universo digital foi liberado para uso público — e, como você deve imaginar, não surgiu com sites sofisticados. A ideia inicial era apenas comunicar, transmitir e coletar dados, e a tecnologia disponível na época oferecia muito menos possibilidades do que temos hoje. As páginas eram, no seu formato comum, pretas no branco, com textos longos e serifados, e links que levavam a outras páginas igualmente simples. Como em qualquer produto, a personalização sempre foi um aspecto relevante tanto para pessoas quanto para marcas.

Por essa razão, cinco anos após o lançamento público da Web, em 1996, foi criada a linguagem de estilo CSS (*Cascading Style Sheets*), ampliando significativamente as possibilidades de personalização de um site. Com ela, tornou-se possível não apenas melhorar a formatação e diferenciar páginas entre si, mas também criar identidades visuais próprias.

Como em todo começo, existe uma fase de exploração e adaptação. É preciso verificar as possibilidades, ir até os limites — ou além deles — e, quando necessário, se destacar. Com a Web não foi diferente. À medida que o número de usuários crescia e o acesso à internet se tornava mais amplo, marcas, universidades e veículos de comunicação começaram a embarcar nessa nova era, dando início

também à competição entre usuários. Certos comportamentos e fluxos precisavam ser aprimorados e simplificados para manter a fidelidade do público e, acima de tudo, garantir uma boa experiência.

Com a intenção de aperfeiçoar produtos digitais, detalhes que normalmente tinham relação com usabilidade e estética visual precisaram ser revistos para estabelecer padrões e consistência. Isso começou a gerar diretrizes sobre como um produto poderia atender melhor às necessidades dos usuários, como as Heurísticas de Usabilidade para User Interface, desenvolvidas por Jakob Nielsen, em 1994.

Nessa época, o termo "UX" não era tão popular quanto hoje, mas muitas empresas passaram a se importar mais com a forma como as pessoas utilizavam os sites, buscando tornar os usuários fiéis à marca. Aos poucos, surgiram diversos padrões de comportamento, já que muitos elementos digitais se inspiravam em objetos do cotidiano, facilitando o reconhecimento e contribuindo para uma melhor usabilidade.

Essa necessidade de criar produtos práticos e fáceis de usar levou muitas empresas a elaborarem modelos de desenvolvimento mais ágeis, como é o caso do Yahoo!, que em 2006 lançou uma biblioteca CSS e JavaScript *open source* chamada de **The Yahoo! User Interface**, contendo componentes e controles para a construção dos seus próprios sites. Historicamente, podemos dizer que esse foi o primeiro design system criado por uma empresa grande. Embora tenha sido concebido principalmente para programação, já que foi construído pelo time de front-end do Yahoo!, o YUI conta com uma documentação com guias de uso, aplicação, exemplos, uma seção de comunidade para colaboração e, naturalmente, linhas de código para utilizar seus elementos.

Digo tudo isso no presente porque a biblioteca ainda está disponível, tanto para leitura quanto para download, representando um marco importante para compreendermos como a evolução foi grande, mas mantendo padrões que vêm desde aquela época.

É possível notar que, embora se trate de um modelo semelhante aos design systems atuais — com estilos, componentes e documentação —, essa nomenclatura sequer existia na época, e o conjunto era apenas considerado e referido como framework.

O aumento considerável de empresas criando suas próprias bibliotecas fez com que, em 2013, esse nome começasse a se popularizar a partir de uma metodologia extremamente famosa e que colaborou ativamente para o levantamento e evolução de muitos design systems. Esse conceito é o **Atomic Design** — metodologia criada pelo designer, consultor, palestrante e escritor Brad Frost.

Basicamente, Frost criou esse processo com o objetivo de simplificar a criação de um design system, fazendo uma assimilação da composição de interfaces com elementos da química e biologia, passando por cinco etapas: **átomos**, **moléculas**, **organismos**, **templates** e **páginas**.

Apesar de causar estranheza para quem não conhece, a integração do Atomic Design na época em que foi pensado colaborou muito não só para promover a consistência visual e funcional, mas também para dar um direcionamento sobre como elaborar um guia de estilo de forma eficaz.

É surpreendente olhar para trás e perceber como tudo evoluiu em prol da consistência, da harmonia e da facilidade de aplicação. Hoje, o design system se tornou um conceito sólido e abrangente, voltado não apenas para aplicações web e apps, mas também para identidade visual, ilustrações, ícones, e-mails e até redes sociais. O que nos aguarda no futuro?

Capítulo 2
Solucionando problemas

Com tudo o que você leu até aqui, é impossível negar que o design system é um produto que consegue trazer uma série de benefícios para uma empresa. Eu não costumo medir esforços para elogiar e dizer o quanto eu gosto de atuar com isso, mas é preciso ser "pé no chão" e entender que um design system não é a solução para tudo, e que pode não ser a escolha certa para cenários diferentes.

Um erro comum é acreditar que o design system é um produto que atua diretamente na solução de problemas de experiência da pessoa usuária. Podemos afirmar com absoluta certeza que um design system facilita a aplicação de boas práticas de UX, mas ele não substitui processos de pesquisa ou validação, nem resolve problemas de usabilidade, arquitetura e qualquer outro detalhe que tenha relação com interação.

Dessa forma, quando falamos de construir, por exemplo, um novo site para o ecossistema de uma empresa, o papel do design system é manter a consistência e tornar o produto escalável. Enquanto houver diretrizes claras que indiquem como um produto deve ser construído corretamente para a marca em questão, fornecendo todos os recursos necessários para que isso possa acontecer, mais rápido e fácil será desenvolvê-lo. Não se trata de resolver a experiência do usuário, mas, se o contexto da sua empresa se enquadra em um ou mais dos itens a seguir, considere criar um design system.

2.1 Problemas que um design system resolve

Inconsistência visual

É normal que, ao acessarmos a página inicial de um site e navegarmos para outras páginas, alguns elementos sejam idênticos ao longo de todo o fluxo — como as margens laterais, as cores e formatos de botões, o cabeçalho fixo no topo, entre outros recursos que reforçam a sensação de estarmos no mesmo site acessado minutos antes. No entanto, convenhamos, nem todos os produtos são assim. Em sistemas muito extensos, por exemplo, é frequente encontrar diversas inconsistências visuais que, dependendo de onde ocorram, podem até fazer com que o usuário se perca durante a navegação.

Em algumas consultorias que realizei antes de implementar um design system, cheguei a encontrar 12 versões de um botão — nenhum era igual ao outro. Havia variação nos tamanhos e espaçamentos, alguns tinham bordas levemente arredondadas, outros eram completamente circulares. Até o texto apresentava inconsistências: ora com apenas a primeira letra maiúscula, ora totalmente em caixa alta.

Ter várias versões de um mesmo componente gera confusão tanto para os usuários quanto para a própria equipe de design e desenvolvimento. A consequência disso é a perda de consistência visual e um aumento no trabalho de criação e manutenção do produto.

Ter um design system para casos como esse permite estabelecer definições claras para a estética visual de uma interface, garantindo que todos os elementos tenham uma aparência consistente ao longo de todo o fluxo, ajudando a criar uma identidade coesa para a marca.

Atraso e demoras no desenvolvimento

A última coisa que queremos na construção de um produto é ter que repetir processos já realizados, tanto na área de UI quanto na de programação. Quase todos que trabalham com produtos já

enfrentaram, ao menos uma vez, prazos de entrega 'para ontem', ou seja, tarefas que precisam ser concluídas rapidamente. Por isso, é fundamental reservar tempo suficiente para testes e validações.

Com um design system em mãos, não precisamos criar elementos do zero sempre que iniciamos algo novo. Designers e desenvolvedores têm acesso a estilos padronizados e bibliotecas de componentes completos, podendo simplesmente reutilizar o que já existe. Isso acelera o desenvolvimento e economiza um grande esforço.

Dificuldade em manutenções

Quando falamos de atualizar ou corrigir algo de um único produto, podemos dizer que essa é uma tarefa, de forma geral, simples, pois o que precisa ser mudado será feito diretamente nas páginas específicas. Mas, quando essa atualização afeta dois ou mais produtos que compartilham os mesmos elementos — ou até apresentam bugs nas mesmas áreas —, a complexidade pode aumentar. Isso é especialmente verdadeiro para quem trabalha no código, já que nem sempre um simples "ctrl+c" resolve. Para situações como essa, um design system se encaixa perfeitamente.

Quando um ou mais produtos são alimentados por um design system, qualquer problema deve ser corrigido diretamente no design system, e não no produto individual. Assim, bugs, falhas ou outras questões são resolvidos de forma imediata, evitando retrabalho.

2.2 Quando um design system não é necessário

Sabendo das enormes vantagens de um design system, você vai querer levá-lo para a sua empresa. Talvez já esteja tentando vender

a ideia para colegas e gestores, acreditando que isso poderá melhorar sua vida e a de todos, certo?

Entretanto, há um equívoco que algumas pessoas cometem: focar totalmente nos benefícios e esquecer dos reais problemas que o design system ajuda a solucionar.

Ter um design system, com certeza, traz pontos positivos. No entanto, ainda existem alguns contextos em que sua implementação pode ser mais complexa, ou até mesmo desnecessária no momento.

Confira alguns dos cenários a seguir:

Projetos pequenos ou de curto prazo

Sabendo que o design system é um produto escalável — como já mencionei e voltarei a mencionar ao longo dos capítulos —, você deve compreender que não faz sentido criá-lo para um site institucional de pequeno porte. Para produtos com início, meio e fim bem definidos, um guia de estilo simples, elaborado diretamente na ferramenta de prototipagem, é suficiente. Esse guia reúne os elementos visuais e componentes básicos a serem utilizados e ainda serve como documento de *handoff* para a pessoa desenvolvedora.

Equipes pequenas e recursos limitados

Design systems são como "filhos recém-nascidos": precisam de atenção constante, pois são produtos escaláveis que necessitam de revisões e atualizações periódicas. Para isso, precisamos de pessoas responsáveis que mantenham essa constância, além de realizar testes para verificar possíveis bugs ou até mesmo problemas com acessibilidade.

Se a sua empresa está passando por um momento no qual se torna mais difícil focar em um produto novo, talvez criar um design system não seja a opção ideal. Nesta ocasião, recomendo a utilização de um UI framework open-source, como o Chakra UI, que é gratuito e fornece não apenas o código para quem desenvolve em front-end

como também um arquivo no Figma para trabalhar no design de interface.

Projetos experimentais

Penso que este seja o item mais óbvio da lista, mas vale reforçar. Um projeto experimental, por definição, pode — se tudo sair como planejado — dar certo e se tornar um produto oficial, recebendo mais atenção da empresa. Por outro lado, as coisas podem não sair como o esperado e o projeto fracassar. Se o único produto que você possui ainda é um experimento, criar um design system em paralelo dificilmente será a melhor escolha, pois pode representar tempo desperdiçado, especialmente em um momento em que o foco deveria estar em outras prioridades.

Agora que você tem em mente os cenários em que um design system pode ou não se tornar um produto de sucesso, vamos iniciar o planejamento, identificando tudo o que precisa ser levantado — desde as necessidades iniciais até a primeira entrega.

Parte 2 - Planejamentos iniciais

Um design system eficaz geralmente nasce da necessidade de eliminar inconsistências e melhorar a eficiência da comunicação visual e funcional entre os produtos de uma empresa. Quando as equipes enfrentam desafios como retrabalho, dificuldade na colaboração e experiências de usuário fragmentadas, torna-se evidente a importância de um planejamento estruturado que integre todos os aspectos essenciais para a implementação do sistema.

Nesta segunda parte do livro, você lerá sobre:

- Os tipos de equipe necessários para implementar um design system.
- A importância do inventário de interface no planejamento inicial.
- Ferramentas e tecnologias essenciais para o desenvolvimento do sistema.

CAPÍTULO 3
Primeiras definições

Ainda estamos no início da nossa jornada sobre design systems, mas imagino que você já saiba que nenhum deles surge apenas pelo simples prazer de ter um. É preciso uma motivação verdadeira, um problema concreto a ser resolvido, para que o produto realmente seja eficiente.

O ponto que pode começar a gerar dúvidas é: quais os primeiros passos que precisam ser dados para começar a construção de um design system? Se você pensou em abrir o Figma e começar a criar seus elementos e componentes, você pensou errado. Como já ressaltei, o guia de estilo é importante, mas se um design system está nascendo, há outras definições que precisam ser tomadas antes de qualquer descrição de design ou tecnologia.

Como o design system é um produto que estará ativo constantemente na rotina de trabalho de outros colaboradores da sua empresa, é extremamente importante entender a percepção que essas pessoas possuem a respeito do modelo atual do desenvolvimento e da manutenção dos produtos, tanto pela visão de design quanto pela de programação. Para que isso possa ser feito de forma prática e bem-sucedida, uma pesquisa quantitativa e qualitativa é sempre bem-vinda.

Uma abordagem combinada de pesquisa quantitativa e qualitativa é valiosa para compreender o cenário e o modelo de trabalho atuais. A pesquisa quantitativa permite medir resultados de maneira objetiva e coletar dados estatísticos sobre métricas específicas, enquanto a pesquisa qualitativa oferece *insights* detalhados e contextuais sobre as experiências, percepções e motivações, enriquecendo a compreensão dos fatores subjacentes que afetam a eficiência do produto.

Ambos os modelos, atuando de forma complementar, poderão proporcionar uma visão holística e aprofundada da eficácia que um design system fornece, permitindo decisões mais assertivas.

Talvez nem fosse necessário dizer que não existe uma fórmula que indique quais as perguntas exatas que você deve fazer nesta pesquisa, já que cada empresa possui um cenário diferente. Entretanto, separei algumas perguntas para ajudá-lo(a) nesse processo:

Perguntas quantitativas

1. Como você descreveria o processo atual de desenvolvimento e manutenção de produtos em nossa empresa?
2. Quão bem você acredita que o modelo atual atende às necessidades dos usuários finais?
3. Como você avalia a comunicação entre equipes de desenvolvimento e design?

4. Quão eficaz você acha que é a colaboração entre equipes de desenvolvimento e design?

Perguntas qualitativas

1. Descreva sua experiência atual com o modelo de desenvolvimento e manutenção de produtos em nossa empresa.
2. Qual ou quais são os principais desafios que você enfrenta no dia a dia relacionados ao modelo atual de desenvolvimento e manutenção dos produtos?
3. Se você pudesse sugerir melhorias ou mudanças específicas no modelo atual de desenvolvimento e manutenção de produtos, quais seriam?
4. Há algum comentário adicional ou experiência que você gostaria de compartilhar em relação ao modelo de desenvolvimento e manutenção de produtos?

Essas são apenas algumas perguntas, elaboradas de forma genérica, que podem se adaptar bem à realidade da sua empresa. Mesmo que apenas as questões quantitativas já ofereçam insights sobre a eficiência do modelo atual, compreender a percepção individual dos colaboradores enriquece muito mais a pesquisa.

Vamos considerar que a etapa de realização da pesquisa tem o objetivo de nos certificar de que um design system pode realmente fazer a diferença, e de que os problemas que esse produto resolve — como vimos no final do capítulo anterior — estão presentes no dia a dia dos colaboradores da sua empresa. De qualquer forma, esse é um processo que será realizado antes das primeiras definições do produto.

Assim que você coletar e analisar os dados, e perceber que o design system será um diferencial positivo, será preciso começar a pensar em algumas outras definições antes de realmente colocar a mão na

massa; isso inclui definir: as pessoas responsáveis pelo design system, a estrutura atual dos produtos, tecnologia e os primeiros passos a serem tomados.

Capítulo 4
Equipe

Por ser um produto extremamente grande e complexo, que só tende a crescer, empresas acabaram notando a necessidade de ter uma equipe focada especificamente em escalar um design system.

Idealmente, um setor multidisciplinar é o caminho correto para a construção de um novo sistema, pois a distribuição de tarefas poderá ser mais bem equilibrada, considerando que essa será a mesma equipe que futuramente dará suporte para todos os outros times da mesma empresa que utilizam o design system ativamente.

Uma das maiores preocupações para a gestão e a direção da empresa é a necessidade de contratar novas pessoas. No entanto, essa decisão pode se tornar complicada devido a prioridades financeiras da empresa. Outra questão é a dificuldade em realocar membros do grupo já existente para um novo produto, o que pode não ser a melhor estratégia.

Há várias maneiras de organizar e estruturar um time de design system, mas há três modelos que quero destacar: **solitário**, **federado** e **centralizado**.

4.1 Solitário

O modelo solitário é caracterizado pela presença de uma pessoa designer e uma pessoa desenvolvedora durante todo o processo de criação e manutenção do design system. Esse modelo é comum e pode ser uma boa porta de entrada principalmente para empresas menores, em que a equipe de design e desenvolvimento é enxuta e

não há muitas demandas em relação à produção de interfaces e produtos digitais.

Nesse modelo, o(a) designer responsável tem a liberdade para criar e gerenciar todas as diretrizes, componentes e padrões visuais e de interação da empresa. Além disso, deve documentar e disponibilizar essas informações para outros designers e desenvolvedores da organização. Isso garante que todos estejam alinhados e trabalhem seguindo os mesmos procedimentos. A principal vantagem é a centralização de tudo o que será construído, facilitando o gerenciamento e a manutenção do produto, já que não há necessidade de passar por diversas mãos.

Por outro lado, esse modelo também apresenta algumas desvantagens, como a possibilidade de os responsáveis ficarem sobrecarregadas com a demanda de criação e manutenção do design system, o que pode afetar a qualidade do trabalho e a eficiência do processo. Além disso, caso um dos membros desse time saia da empresa, pode haver dificuldades na continuidade do trabalho, já que determinadas informações e diretrizes do design system ficariam concentradas em uma única pessoa.

Portanto, o modelo solitário é viável para empresas menores, mas exige boa gestão do processo e da documentação para garantir qualidade e continuidade.

4.2 Federado

O modelo federado é caracterizado por ter um grupo distribuído por diferentes áreas de uma empresa, cada uma com sua própria expertise e foco. Essas equipes colaboram de forma descentralizada, compartilhando conhecimentos e recursos através de um canal comum de comunicação.

Essa abordagem é ideal para empresas grandes e complexas, que possuem várias unidades de negócio com necessidades e objetivos distintos. Cada time federado é responsável por desenvolver e manter os componentes do design system relevantes para sua área. Isso significa que cada equipe pode ter sua própria linguagem visual, interações e fluxos específicos.

Para que esse modelo funcione bem, é necessário estabelecer uma liderança forte e uma estrutura de governança clara, que coordene a colaboração e a comunicação. É fundamental que a liderança de governança estabeleça padrões comuns e objetivos compartilhados para garantir a consistência e a integridade do design system como um todo, já que será difícil um momento em que todas as pessoas possam participar de reuniões ou dinâmicas de forma síncrona.

Uma das principais vantagens desse modelo é que ele permite uma maior flexibilidade e adaptabilidade às necessidades específicas de cada área da empresa. Além disso, a descentralização do design system pode incentivar a inovação e a experimentação em diferentes times da organização, já que cada equipe pode testar e iterar seus próprios componentes.

Contudo, a descentralização pode comprometer a consistência da experiência da pessoa usuária se os setores não estiverem bem coordenados. Por isso, padrões claros de colaboração e governança são essenciais para garantir a integridade do design system federado.

4.3 Centralizado

O modelo de time centralizado é um dos modelos mais comuns em grandes empresas e envolve a criação de uma equipe dedicada

exclusivamente ao design system, responsável por sua manutenção e atualização constante.

Nesse modelo, o time centralizado é responsável por criar e manter as diretrizes e documentação do sistema, além de desenvolver e atualizar seus componentes. As pessoas trabalham de forma colaborativa com outros times de design e desenvolvimento que utilizam o design system em seus projetos.

A vantagem desse modelo é que ele garante consistência e qualidade em todo o sistema, além de promover a reutilização de componentes e reduzir o tempo de desenvolvimento. No entanto, pode haver dificuldades em manter a comunicação e colaboração entre os diferentes times envolvidos no projeto.

Para que esse modelo seja bem-sucedido, é necessário um forte comprometimento do setor de design system com os demais times da empresa. Isso inclui ouvir suas necessidades e tornar o produto cada vez mais colaborativo. Além desse fator, é necessário que haja um processo bem definido para a integração do design system em novos projetos, contando com o feedback constante das equipes.

4.4 Design System Ops

Tendo explorado os diferentes modelos e como cada um impacta a evolução de um design system, chegamos ao ponto crucial sobre como operacionalizar esse produto no dia a dia. É aqui que entra um termo muito conhecido e que talvez você já tenha encontrado em algum lugar: Design System Ops.

Design System Ops é a área responsável por garantir que um design system funcione de maneira eficiente dentro de uma organização. Ela envolve a definição de processos, ferramentas e fluxos de trabalho para que designers e desenvolvedores possam utilizá-lo, mantê-lo e evoluí-lo sem atritos. Em vez de focar apenas na criação

de componentes visuais ou na implementação de código, o Design System Ops trabalha para estruturar e operacionalizar todo o ecossistema do design system, garantindo que ele seja sustentável e escalável.

O Design System Ops atua como a ponte entre equipes, garantindo que o design system esteja integrado ao fluxo de trabalho da empresa. Ele lida com governança, automação, documentação e comunicação entre times, ajudando a evitar inconsistências e promovendo o uso eficiente dos recursos. Além disso, ele também facilita a adoção do design system, criando guias, treinamentos e processos para que todos saibam como utilizar e contribuir para a sua evolução.

No dia a dia, isso envolve a automação de processos para a publicação de novos componentes, a definição de estratégias para versionamento e controle de mudanças, além do acompanhamento do uso do design system. Também inclui o suporte contínuo à comunidade interna de usuários. Quando bem estruturado, o Design System Ops torna o sistema um recurso dinâmico, promovendo eficiência na empresa e garantindo uma experiência do usuário consistente e de alta qualidade.

Capítulo 5
Auditoria de UI

Se o objetivo é construir um design system, é necessário entender quais são os elementos e componentes que precisamos priorizar, como também as inconsistências de interface existentes no produto. Uma das formas de levantar esses cenários é realizando uma **auditoria de UI**.

Uma auditoria de UI é uma avaliação detalhada e sistemática da consistência visual, usabilidade, fluxo de interação e quaisquer outras características de uma interface de usuário. Ela é realizada para identificar problemas, inconsistências ou oportunidades de melhoria ao longo das páginas de um produto, e garantir que ela atenda às necessidades e expectativas dos usuários.

Durante a realização de uma auditoria, o responsável, que de preferência tenha conhecimentos intermediários ou avançados em UI Design, analisa todos os aspectos da interface, incluindo os elementos visuais, como cores e tipografia, componentes, interações e microinterações. O objetivo é identificar problemas que possam afetar a usabilidade, a acessibilidade, a eficiência e a experiência do usuário.

Aplicando uma auditoria

Inventário de Interfaces

Antes de começar uma auditoria, é importante fazer um levantamento de tudo o que já existe na interface dos produtos por meio de um inventário de interfaces. Essa é uma dinâmica simples de ser feita, embora possa ser demorada conforme a quantidade de páginas que o produto que você estará inventariando pode ter.

Basta passar cada página do produto e tirar capturas de tela dos estilos e componentes que você for identificando ao longo dessa dinâmica. Se o seu cenário envolver sites, lembre-se de validar também a adaptação para dispositivos menores, pois há uma série de interfaces que se adaptam conforme as resoluções aumentam e diminuem.

Tratando inicialmente dos estilos, você pode considerar tudo o que envolve cor, tipografia, sombra, formas e outros elementos básicos de uma interface, lembrando que você não precisa registrar várias vezes um mesmo elemento que está em diferentes páginas. Considere apenas uma única captura de tela e, se possível, anote características como hexadecimais e detalhes de texto (como tamanho, altura de linha e peso).

Conforme você for identificando componentes, é importante capturar não só o seu estado natural, como também as interações que ele pode ter. Por exemplo: um *checkbox* marcado e desmarcado, efeitos *hover*, *active* e *focus*, e até mesmo versões desabilitadas.

A principal vantagem de levantar esse inventário é ter a noção do escopo de criação do design system. Você só vai trabalhar em cima dos elementos que estão fazendo parte do contexto dos produtos. Alguns são extremamente comuns de se encontrar em qualquer produto digital, como é o caso dos botões, *inputs* e *selects*. Tudo o que não for identificado nesse levantamento não precisa ser considerado para o lançamento da primeira versão do design system.

Eu percebo que algumas pessoas gostam de se "precaver" nesse cenário, criando coisas que ainda não possuem caso de uso. Por exemplo, se em nenhum local você estiver utilizando um componente de *switch*, então não há motivo para criá-lo, até que se torne uma necessidade.

Para organizar suas capturas de tela, utilize uma ferramenta com a qual você possa inserir as imagens e organizá-las de forma

categorizada. Recomendo ferramentas de *whiteboard*, como FigJam e Miro. Não há a necessidade de fazer anotações ou outras considerações, apenas realizar uma análise geral de tudo que foi mapeado:

- Apresentar os elementos visuais, como as paletas de cores do produto, bem como suas variações de tonalidade, escalas tipográficas, níveis de sombra... e contabilizar cada elemento individualmente;
- Contabilizar a quantidade de componentes identificados e, para aqueles que possuem inconsistências, a quantidade de variações distintas.

Um outro ponto que pode ser valioso para essa dinâmica, principalmente para os componentes, é apresentar quantas vezes ele aparece no produto. Isso ajudará a classificar a ordem de prioridade de design e desenvolvimento, colocando aqueles com maior uso no topo da lista.

Como eu já citei, esse é um processo que pode ser demorado. Informações podem ser despercebidas ao longo de várias páginas. Para facilitar o seu processo quanto aos estilos utilizados, a ferramenta **CSS Stats** (https://cssstats.com/) pode ser uma mão na roda. Ao informar o endereço do site, a plataforma retorna com uma análise completa dos estilos, mostrando as cores em preenchimentos e bordas, imagens de plano de fundo, sombras, arredondamento das bordas, escalas tipográficas e vários outros detalhes que você mesmo pode conferir.

Avaliando e tomando decisões

Enquanto o inventário é apenas uma etapa de reunião de informações, a auditoria é o momento em que cada elemento capturado será analisado individualmente. O objetivo é identificar problemas e buscar a melhor forma de solucioná-los no design system.

É nessa fase que encontramos uma série de inconsistências visuais no produto, como:

- A mesma cor, mas aplicada com variações de tonalidade quase imperceptíveis.
- Componentes com tamanhos, espaçamentos e outras propriedades distintas, formando uma série de variações incompatíveis.
- Formas de escrita aplicadas de maneiras diferentes, como cada palavra iniciando com letra maiúscula em alguns lugares, mas em outros, não.
- Ícones de diferentes bibliotecas e estilos visuais aplicados em conjunto.
- Fontes diferentes em componentes iguais.

Figura 5.1: Imagem de um inventário de botões de interface em fundo branco, mostrando diversas variações visuais do componente botão. Há diferenças nas cores (roxo, vermelho, laranja, cinza), tamanhos, bordas (retas e arredondadas), estilos (cheio, contornado e texto), e uso de caixa alta ou não. Os textos incluem "Salvar e publicar", "Próximo", "Cancelar", "CONTINUAR", entre outros, evidenciando inconsistências de design.

Esses e vários outros são resultados comuns quando não se usa um design system ou simplesmente um *framework*, pois sempre que um objeto precisa ser usado novamente ele é construído do zero, perdendo as referências dos outros já criados.

Além de identificar as diferenças que existem entre os elementos, é necessário tomar a decisão de como seguir com a padronização deles. Seus critérios de decisão podem se basear em aspectos como consistência visual, acessibilidade, escalabilidade e viabilidade técnica.

Nesse estágio, trazer mais designers e programadores para validar o impacto da experiência e do aspecto técnico pode ser enriquecedor, como também trazer Product Managers ou outras pessoas gestoras para entender as prioridades de negócio.

A auditoria não é o momento de colocar a mão na massa. As decisões tomadas aqui podem ser ajustadas futuramente, conforme estudos mais aprofundados que você realizar. Com o resultado da auditoria, você terá acesso a uma série de informações, como:

- Quantidade de problemas encontrados;
- Lista de componentes que precisam ser criados;
- Elementos que precisam ser padronizados;
- Prioridades de criação.

Com isso, você e seu time terão um direcionamento sobre como começar a construção do guia de estilo do design system.

Capítulo 6
Escolhendo peças, produtos e pessoas

Existem inúmeras maneiras de definir prioridades na construção de um novo produto. Quando se trata de um design system, essa estratégia deve ser feita de modo que todos se sintam seguros, evitando impactos negativos na entrega e no uso dos produtos.

Para evitar problemas e estabelecer como começar a construção de um design system, você pode usar uma dinâmica eficiente chamada *Picking Parts, Products & People* (Escolhendo Peças, Produtos e Pessoas), escrita por Nathan Curtis e publicada no Medium: https://medium.com/eightshapes-llc/picking-parts-products-people-a06721e81742.

Organizando a dinâmica

Essa dinâmica deve ser realizada em equipe. Por isso, organize e convide todas as pessoas que tenham contato com o design system. Reserve pelo menos 1h30min na agenda dos participantes: utilize os primeiros 30 minutos para explicar a atividade e formar as equipes e a hora seguinte para a aplicação prática.

A dinâmica ocorre em duas etapas: a primeira, intitulada 'Escolhendo as peças', e a segunda, 'Priorizando o sistema'. Cada participante deverá receber um documento com duas páginas — 'Peças' e 'Sistema' —, sendo cada uma utilizada exclusivamente em sua respectiva etapa.

Etapa 1: Escolhendo as peças

Distribua a primeira página do documento para cada participante. Oriente que cada um deve:

1. Riscar ou fazer um “X” nas categorias que julgarem irrelevantes.
2. Circular as cinco categorias que julgarem ser as mais importantes.
3. Marcar as 25 peças que consideram ser mais importantes. Cada peça é composta por itens (por exemplo: Color) e subitens (por exemplo: Contrast) das listas.
4. Após definirem as 25 peças, peçam aos participantes que escolham e circulem apenas as 10 que considerarem mais importantes.

Assim que todos tiverem finalizado a dinâmica, promova uma discussão sobre o que preencheram. Nesse momento, todos devem compartilhar suas ideias a respeito do que consideram como importante ou irrelevante.

Você pode experimentar começar buscando pelas informações similares preenchidas e ir variando os tópicos, puxando até mesmo as categorias menos escolhidas entre todas.

PARTS

NAME _____ DATE _____

VISUAL LANGUAGE

- ☐ Color
  - ☐ Palettes, by category
  - ☐ Contrast
  - ☐ Meaning
  - ☐ Swatches
- ☐ Typography
  - ☐ Hierarchy
  - ☐ Weights / types
  - ☐ Web fonts
  - ☐ Baseline grid
- ☐ Iconography
  - ☐ Pictograms
  - ☐ Illustration
  - ☐ Icon fonts
  - ☐ Alignment
- ☐ Motion
  - ☐ Principles
  - ☐ Timing
  - ☐ Types
  - ☐ Transitions
- ☐ Space
  - ☐ Units / measurments
  - ☐ Metrics and keylines
  - ☐ Structure
- ☐ Photography

UI ELEMENTS

- ☐ Paragraph
- ☐ Lead paragraph
- ☐ Block quote
- ☐ Headers (H1 - H6)
- ☐ Lists
- ☐ Links
- ☐ Code
- ☐ Buttons
  - ☐ Primary
  - ☐ Secondary
  - ☐ Button groups
  - ☐ Menu buttons
  - ☐ Split buttons
  - ☐ FAB (floating action)
- ☐ Form controls
  - ☐ Text
  - ☐ Variants (email, etc)
  - ☐ Text area
  - ☐ Radio
  - ☐ Checkbox
  - ☐ Select

Form controls, cont'd

- ☐ File upload
- ☐ Labels
- ☐ Floating labels
- ☐ Microcopy
- ☐ Required
- ☐ Validation
- ☐ Divider / rule
- ☐ Switch
- ☐ Slider
- ☐ Image
  - ☐ Block
  - ☐ Full bleed
  - ☐ Inline with positions

UI COMPONENTS

- ☐ Action bar/sheet
- ☐ Back to top
- ☐ Badges
- ☐ Breadcrumbs
- ☐ Calendar picker
- ☐ Caption
- ☐ Cards
- ☐ Chat (live)
- ☐ Code block
- ☐ Comments
- ☐ Contact us
- ☐ Cookie notification
- ☐ Data tables
- ☐ Dialog
- ☐ Drawer/accordion
- ☐ Filters
- ☐ Footer
- ☐ Footnotes
- ☐ Header
- ☐ Hero billboard
- ☐ Hero carousel
- ☐ Legend
- ☐ Loading / spinner
- ☐ Local navigation
- ☐ Maps
- ☐ Menu
- ☐ Messaging
  - ☐ Toast
  - ☐ Block
  - ☐ Inline error
- ☐ Modal
- ☐ Nav Megamenu
- ☐ Pagination
- ☐ Progress bar
- ☐ Pull quote
- ☐ Ratings
- ☐ Reviews
- ☐ Rich text editor
- ☐ Search
- ☐ Sidebar
- ☐ Site/App navigation
- ☐ Social networking
- ☐ Status (alpha, beta,...)
- ☐ Tabs
- ☐ Tags / chips / pills
- ☐ Thumbnails / #-ups
- ☐ Tips / tutorial
- ☐ Toolbar
- ☐ Tooltips
- ☐ Video & media

UI PATTERNS

- ☐ Authentication
- ☐ Create account
- ☐ Database connection
- ☐ Form structure
- ☐ Launch
- ☐ Permissions
- ☐ Purchase/checkout
- ☐ Settings
- ☐ Site / app structure
- ☐ Swipe to refresh

PAGE TEMPLATES

- ☐ Home
- ☐ Category
- ☐ Product
- ☐ Dashboard
- ☐ Article
- ☐ Search results
- ☐ Contact us
- ☐ Getting started
- ☐ Discussion
- ☐ Gallery

LAYOUT SYSTEMS

- ☐ Grids for sites
- ☐ Grids for apps
- ☐ Paneling/transitions
- ☐ Responsive/breakpoints
- ☐ Screen sizes

EDITORIAL

- ☐ Voice & tone
- ☐ Writing for the web
- ☐ Word list
- ☐ Capitalization
- ☐ Punctuation

BRANDING

- ☐ Identity
- ☐ Logo(s)
- ☐ Tagline(s)
- ☐ Partnerships
- ☐ Themes

RESEARCH & USERS

- ☐ User needs
- ☐ Personas
- ☐ Research techniques
- ☐ Surveys
- ☐ A/B, multivar. testing

PRACTICES

- ☐ Accessibility
- ☐ Prototyping
- ☐ Performance
- ☐ SEO
- ☐ Design thinking

CODING STANDARDS

- ☐ Style
- ☐ Browsers & devices
- ☐ Environments
- ☐ Prog. enhancement
- ☐ Version control

DOWNLOADS

- ☐ Templates (PSD,...)
- ☐ Swatch palettes
- ☐ Fonts
- ☐ Icons
- ☐ Code

GETTING INVOLVED

- ☐ Request New Parts
- ☐ Give Feedback
- ☐ Request Audit
- ☐ Present at a Share
- ☐ Contribute

ABOUT THE PROGRAM

- ☐ Home page
- ☐ Principles / pillars
- ☐ Getting started
- ☐ FAQ
- ☐ Version history

ABOUT THE TEAM

- ☐ Team
- ☐ Recruiting / jobs

UTILITIES

- ☐ Variables / tokens
- ☐ Mixins
- ☐ Helpers
- ☐ Customize

OTHER?

Figura 6.1: Primeira página do documento. No topo está o título Parts, seguido pelos campos a serem preenchidos Nome e Data. Abaixo, diversas áreas categorizadas a serem preenchidas, sendo elas a Visual Language, UI Elements, UI Components, UI Patterns, Page Templates, Layout Systems, Editorial, Branding, Research e Users, Practices, Coding Standards, Downloads, Getting Involved, About the Program, About the Team e Utilities.

Assim que todos terminarem, reúna os documentos em um espaço visível a todos, de modo que cada participante possa observar os resultados dos demais. Em seguida, conduza uma discussão sobre as escolhas realizadas, incentivando que compartilhem seus pensamentos e decisões.

Aproveite também para destacar suas próprias observações sobre o resultado geral, como as categorias mais escolhidas e aquelas que não receberam nenhuma seleção.

Etapa 2: Priorizando o sistema

Finalizadas as discussões, inicie a segunda etapa distribuindo a outra página do documento, que contém três tabelas, cada uma com duas colunas e dez linhas, intituladas "*Parts*", "*Products*" e "*People*".

O objetivo dessa etapa é identificar quais peças cada participante considera prioritárias para o design system, em quais produtos da empresa devem ser aplicadas e quais pessoas devem estar envolvidas nesse processo.

SYSTEM

NAME _____ DATE _____

PARTS

NAME	VALUE
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

PRODUCTS

NAME (TIER)	PLATFORM

Tier: **F** (Flagship) • **S** (Secondary)
Platform(s): **WS** Web Site • **WA** Web App • **iOS** • **AND** Android • **WIN** Windows • [Other]

PEOPLE

NAME	DISCIPLINE(S)

Discipline: **UX** • **DESign** • **CONtent** • **FED** • **Dev** • **QA** • **PdM** • **PjM** • **Exec**

Figura 6.2: Segunda página do documento. No topo está o título System, seguido pelos campos a serem preenchidos Nome e Data. Abaixo, há três blocos: Parts, com duas colunas

e dez linhas; Products, com duas colunas e 20 linhas; e People, também com duas colunas e 20 linhas.

Na primeira tabela (*Parts*), devem ser registradas as 10 peças selecionadas na rodada final da primeira etapa, agora em ordem de prioridade. A primeira linha corresponde à prioridade máxima e a última, à mínima.

Cada pessoa deve informar o nome da peça selecionada e dar um valor para ela, considerando que, ao todo, cada participante tem 25 pontos para distribuir entre elas. Ou seja, se der oito pontos para a primeira peça, sobrarão 17 para distribuir entre as outras nove peças.

Para a segunda tabela (*Products*), os participantes devem anotar, de forma livre, os nomes dos produtos da empresa que o design system deve contemplar, indicando também em quais plataformas operam (Web, Windows, MacOS, Android, iOS etc.). Após a listagem completa, cada participante deve identificar até quatro produtos considerados mais críticos — aqueles que mais necessitam do design system — e até quatro produtos secundários, que serão priorizados em seguida.

Por último, na terceira tabela (*People*), cada um deve escrever o nome de todas as pessoas que devem dedicar tempo à construção do design system, indicando também a área de atuação de cada uma. Em seguida, peça que identifiquem as principais pessoas influentes do time, assim como lideranças mais distantes com quem seja importante alinhar expectativas, como um diretor de marketing.

Assim como na primeira etapa, inicie uma nova discussão sobre todas as anotações feitas pelos participantes. Proponha um momento de fala de 3 a 5 minutos para que cada um compartilhe individualmente uma ou duas impressões sobre a dinâmica. Somente após esse momento, reúna todos os documentos, analise as anotações e identifique as ideias consensuais e as divergentes.

Capítulo 7
Roadmap

Criar o *roadmap* de um design system não é um trabalho difícil, uma vez que ele é um produto. Se você já traçou isso para outros serviços, como sites e aplicativos, pode se sentir em casa: o caminho é o mesmo, as únicas diferenças são nos requisitos e nas entregas.

Para essa definição, falamos apenas da construção do design system, e não da futura inclusão dele em algum dos produtos da empresa. Para isso, é necessário analisar todos os dados levantados e as prioridades identificadas nas dinâmicas realizadas até aqui e, em conjunto com o time, definir quanto tempo deve ser estipulado para lançar a primeira versão, que nada mais é do que um MVP (Produto Mínimo Viável), ou seja, a versão inicial, simples e enxuta de um produto.

Não cabe a mim dizer qual forma de realizar o *roadmap* é a melhor para esse cenário. Cada empresa, time e pessoa produz de formas diferentes. Mas um modelo interessante, que coincidentemente descobri em um curso de design system ministrado pelos designers Bruno Coronel e Guilherme Gonzalez, é o **GO Product Roadmap**, criado pelo especialista em Product Management Roman Pichler.

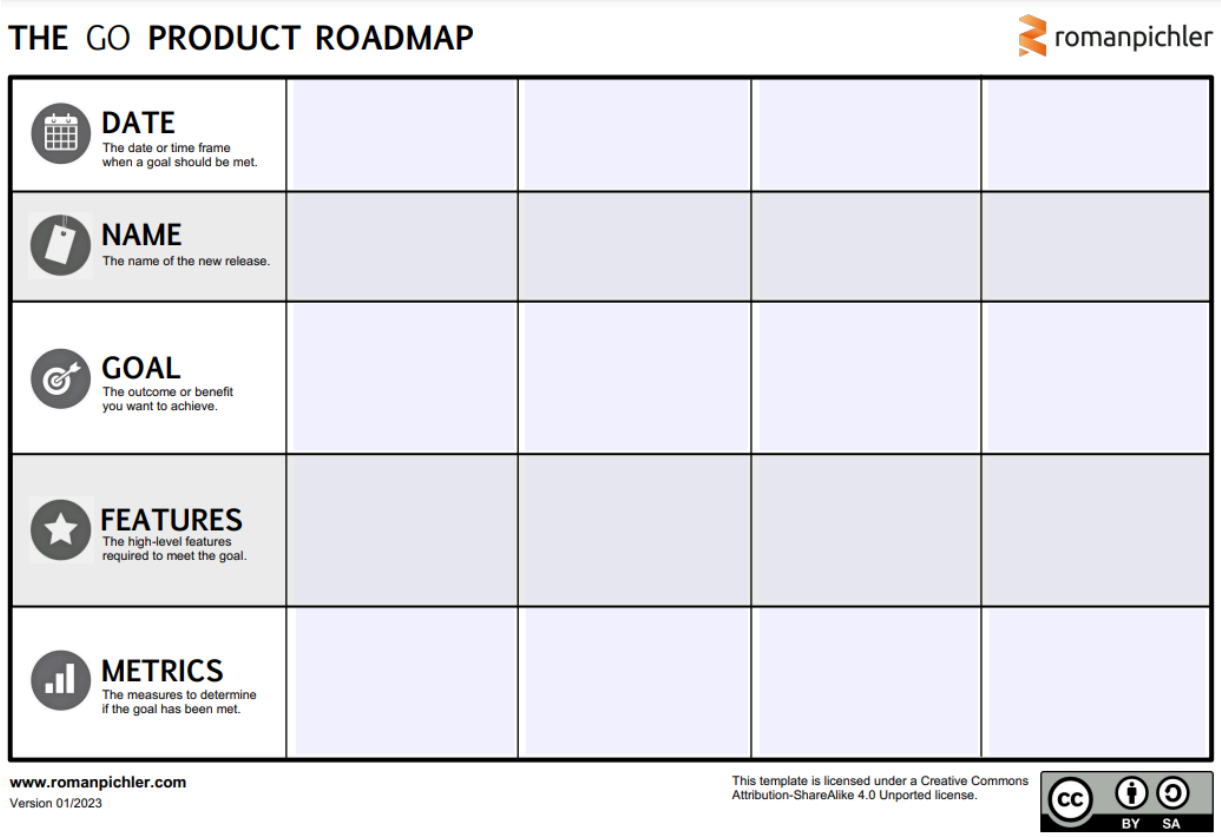


Figura 7.1: Documento do GO Product Roadmap apresenta o título do modelo na parte superior esquerda e, na direita, o logotipo do autor Roman Pichler. Abaixo, há quatro colunas e cinco linhas, sendo, de cima para baixo: Data, Nome, Metas, Funcionalidades e Métricas.

Esse método de *roadmap*, como o próprio autor explica, é orientado a objetivos (ou, em inglês, **g**oal-**o**riented), ou seja, além de definir prazos, é preciso estabelecer metas e planejar o que deverá conter em cada lançamento — não apenas para produtos novos, mas também para promover melhorias em produtos já existentes. Essa ferramenta é baseada em quatro princípios, segundo o próprio autor:

- Foco nos objetivos e metas de produto;
- Priorizar as funcionalidades de produto;
- Definir os marcos do produto;
- Monitorar os progressos do produto e ajustar conforme for necessário.

No plano do *roadmap* é preciso preencher as informações de **data** e **nome** do lançamento da versão, assim como **as metas** que seu time deseja alcançar, as **funcionalidades** e outros detalhes que serão incluídos. Também é necessário registrar as **métricas** que indicarão se você e sua equipe atingiram os objetivos.

Essa forma de visualização permite que qualquer pessoa do time, além de estar alinhada com os entregáveis, consiga visualizar os planos futuros do produto.

Quando relacionamos isso com design systems, o ideal é planejar as metas das primeiras versões com as definições iniciais da biblioteca de um design system, como paletas de cores, escalas tipográficas, grids e breakpoints, para que somente nas próximas versões seja possível iniciar a construção de componentes. Você pode ir além na definição dos lançamentos e detalhar os prazos para as construções no lado de design e no lado da programação. Afinal, embora sejam os mesmos itens, eles são utilizados em ferramentas distintas por pessoas diferentes.

Veja um exemplo de como esse *roadmap* pode ser documentado:

- Data: 30 dias;
- Nome: Componentes de formulário;
- Meta: Recriar e documentar os componentes de formulário seguindo os padrões do design system;
- Funcionalidades: Text field, Textarea, Select, Checkbox, Radio Button, Switch;
- Métricas: seis componentes recriados e documentados.

As métricas que você utilizará para definir o sucesso disso poderão ser baseadas no próprio inventário de interfaces que contém o levantamento de toda a informação utilizada nos produtos da sua empresa.

Como você mesmo pode perceber, o mapa da metodologia possui somente quatro colunas a serem preenchidas, trazendo quatro futuras versões com suas respectivas funcionalidades; contudo, nada impede que você aumente a quantidade de colunas conforme sua necessidade, para traçar o futuro do seu design system a longo prazo.

Entretanto, todo design system de sucesso, além de toda a sua biblioteca com estilos, componentes, ilustrações e tudo o que couber no pacote, ainda possui um item a mais que, do meu ponto de vista, é a principal forma de alinhar tudo o que será construído e escalado futuramente: a **governança**.

Parte 3 - Governança

Governança é um termo muito conhecido no universo do design system, mas que, à primeira vista, pode gerar dúvidas quanto ao seu real significado. Ela se refere a um conjunto de regras, políticas e processos que visam controlar e preservar a integridade do sistema em um nível estratégico, abrangendo desde a forma como o produto é utilizado até definições e comportamentos relacionados à gestão de equipes e responsabilidades.

Na prática, a governança é fundamental para garantir que todos os envolvidos na construção estejam alinhados com os objetivos, os padrões de qualidade e a consistência em longo prazo, evitando que o sistema se torne desorganizado, desatualizado ou pouco útil para as pessoas que vão utilizá-lo. Por isso, o processo de governança deve ser planejado e implementado com cuidado.

Nesta terceira parte do livro, você lerá sobre:

- Como garantir a consistência da marca no design system.
- As responsabilidades da equipe na manutenção e evolução do sistema.
- Estratégias para versionamento e colaboração eficiente.

Capítulo 8
Consistência da marca

A consistência da marca é um aspecto fundamental em qualquer design system, pois ajuda a estabelecer uma identidade visual coesa e reconhecível para uma empresa ou organização. Trata-se de garantir que todos os elementos visuais, desde cores e tipografia até ícones e estilos de interação, sigam um conjunto consistente de diretrizes de design.

Por si só, a consistência não apenas fortalece a identidade da marca, mas também ajuda a criar uma experiência mais coesa e fácil para

os clientes. Quando integrada ao design system, ela facilita a construção e a escalabilidade dos produtos da empresa, mantendo a marca em evidência.

Manter a consistência da marca por meio de um design system traz uma série de benefícios. O mais comum e talvez mais importante é a **usabilidade** dos produtos. Quando uma identidade visual é consistente, os elementos visuais, como cores, tipografia, ícones e layouts, seguem padrões predefinidos com que os usuários já estão familiarizados. Isso torna a experiência mais intuitiva, visto que os usuários sabem o que esperar e onde encontrar as informações desejadas.

Além disso, a consistência ajuda a reduzir a carga cognitiva, pois não é necessário reaprender padrões visuais a cada interação. Como resultado, a usabilidade é aprimorada, proporcionando uma experiência mais eficiente e agradável.

Outro fator que a consistência promove é no **reconhecimento da marca**, permitindo que os usuários identifiquem facilmente os produtos e serviços da empresa.

Essa percepção ocorre principalmente no uso dos elementos que compõem a identidade visual, como cores, tipografias e elementos gráficos. Marcas como McDonald’s e Coca-Cola são referências nesse aspecto, pois, ao longo de suas trajetórias, criaram diversas campanhas que alcançaram um grande público. Por exemplo, ao vermos uma propaganda na televisão durante o período natalino com um urso polar, imediatamente a associamos à Coca-Cola.

Entretanto, uma marca não precisa necessariamente dos elementos visuais para ser reconhecida. A forma como ela se comunica com seus clientes, incluindo efeitos sonoros, pode contribuir para esse reconhecimento. Um dos maiores exemplos atualmente é o “tudum” da Netflix. Ao ler isso, é bem provável que você tenha falado ou pensado a expressão com a mesma entonação em que é tocada no início de um filme ou série, considerando que é um som marcante —

exatamente por isso, está presente em todas as mídias da plataforma. Você não precisa ver o logotipo, o site, o app ou qualquer outro elemento visual para saber que está assistindo na Netflix.

Além disso, uma identidade visual coesa transmite profissionalismo e confiabilidade, gerando confiança nos usuários. A consistência também contribui para uma experiência mais fluida, já que os elementos visuais familiares ajudam na navegação e interação de forma mais intuitiva.

A relação entre a consistência da marca e o design system é direta e interdependente. Um design system bem definido e implementado incorpora diretrizes de consistência da marca em todos os seus componentes e padrões de design.

Isso significa que cores, tipografia, ícones e outros elementos visuais são escolhidos e aplicados de acordo com os princípios e valores da marca. Por sua vez, o design system ajuda a garantir a consistência da marca ao fornecer um conjunto centralizado de recursos e diretrizes acessíveis a toda equipe de design e desenvolvimento.

No entanto, implementar e manter a consistência da marca pode apresentar desafios. É necessário um esforço coordenado entre equipes de design, desenvolvimento e marketing para garantir que todas as partes interessadas estejam alinhadas com as diretrizes da marca.

Para garantir o sucesso, é importante seguir algumas boas práticas ao lidar com a consistência da marca em um design system. Isso inclui:

- Estabelecer diretrizes claras e acessíveis para toda a equipe.
- Realizar revisões regulares nos produtos, a fim de garantir conformidade.
- Flexibilidade para acomodar mudanças quando necessário.

Também é essencial que, além desses fatores, haja uma comunicação aberta e colaborativa entre todas as partes interessadas para garantir a coesão e o alinhamento em relação à identidade da marca.

Em resumo, a consistência da marca desempenha um papel crucial em um design system: ajuda a fortalecer a identidade visual e promove uma experiência de usuário coesa e intuitiva.

Ao incorporar diretrizes de consistência da marca em todo o design system e seguir as melhores práticas de implementação, as empresas garantem que sua marca seja reconhecível, confiável e impactante em todos os pontos de contato com os usuários.

CAPÍTULO 9
Responsabilidades da equipe

O design system é um produto que contempla uma série de fatores e decisões de design. Devido à sua complexidade e evolução contínua, muitas empresas tendem a contratar pessoas para trabalhar exclusivamente com ele.

Na segunda parte do livro, você leu sobre alguns dos modelos que permitem estruturar uma equipe de design system. Para complementar esse assunto e destacar um fator essencial para a governança, vamos abordar aqui as responsabilidades da equipe como um todo e de cada pessoa individualmente.

A importância de definir essas responsabilidades não pode ser subestimada. Ao atribuir papéis claros dentro da equipe, é possível garantir uma colaboração produtiva e uma governança eficiente do sistema.

Isso promove a transparência, a responsabilidade e a prestação de contas dentro do grupo, garantindo que todos os membros estejam alinhados com os objetivos e diretrizes do design system.

A vantagem de estabelecer as responsabilidades da equipe antes mesmo de iniciar a construção da biblioteca é evitar dúvidas ou desentendimentos futuros sobre as funções de cada integrante do time. Uma das formas de fazer isso é utilizando o framework **Matriz RACI**, amplamente conhecido por ajudar a organizar e esclarecer responsabilidades.

A Matriz RACI é uma ferramenta de gestão de projetos bastante utilizada para definir e organizar as responsabilidades dos membros em relação às atividades do projeto. Trata-se de um framework simples e eficaz que ajuda a esclarecer quem é responsável, quem

precisa ser consultado, quem deve ser mantido informado e quem deve aprovar cada tarefa ou atividade específica.

A sigla RACI é derivada de quatro papéis principais: Responsável (*Responsible*), Aprovador (*Accountable*), Consultado (*Consulted*) e Informado (*Informed*). Esses papéis são atribuídos a cada atividade ou tarefa e ajudam a garantir clareza e transparência sobre quem está envolvido em cada aspecto do projeto.

Ao definir claramente as responsabilidades de cada membro da equipe, a Matriz RACI ajuda a evitar mal-entendidos, promove a colaboração eficaz e melhora a eficiência geral do projeto.

Atividades	**Product designer**	**UX Writer**	**Desenvolvedor front-end**
Padrões de design	R	I	I
Documentação	C	R	I
Versionamento	C	I	R

Para não desviar do foco deste livro, recomendo a leitura do artigo *Matriz RACI: o que é e como aplicar em sua gestão* (https://blog.runrun.it/matriz-raci/) do *Runrun.it* (caso deseje aprofundar seu entendimento sobre esse framework simples — porém poderoso).

Não será difícil elaborar a Matriz RACI, mas isso não significa que o processo será rápido. Eu mesmo já participei dessa dinâmica algumas vezes e, quanto maior for o time, maior a chance de as discussões levarem mais tempo, podendo ser necessário marcar até dois encontros para finalizar a atividade. De todo modo, não importa se sua equipe tem 2 ou 20 pessoas. É importante alinhar as expectativas para evitar os problemas de comunicação.

Nem toda empresa ou cenário exige que uma equipe de design system contenha membros de diferentes áreas. Isso não impacta negativamente a construção, evolução e implementação do produto. Posto isso, uma equipe multidisciplinar pode conter os seguintes profissionais:

- Product designers;
- UX Writers;
- Desenvolvedores front-end.

1. Product designer

Os product designers desempenham um papel vital ao definir a visão, estratégia e *roadmap* do design system. Por terem mais contato com outros designers da empresa e, até mesmo, com os próprios usuários, são as pessoas que melhor enxergam as necessidades do produto e problemas que precisam ser resolvidos.

Esses profissionais possuem diversas responsabilidades, considerando que toda novidade ou melhoria do design system precisa ser compreendida e viabilizada pelos product designers antes do início do desenvolvimento. Garantir a consistência da marca (discutida no capítulo anterior), dos princípios de design (tema que será abordado no capítulo 12) e da acessibilidade é fundamental para o funcionamento do sistema, assim como a definição de estilos, design tokens e componentes.

De todo modo, os product designers não devem considerar somente os usuários. São os colaboradores que utilizam o design system ativamente e, por essa razão, em colaboração com o time de desenvolvimento front-end, precisam fornecer um produto que seja fácil de usar.

2. UX Writer

Convenhamos que nem sempre é possível contar com um UX Writer alocado ou disponível para colaborar no desenvolvimento do design

system, mas esses especialistas possuem um conhecimento único e essencial para a evolução desse produto.

UX Writers focam em garantir que a linguagem e a comunicação nos componentes sejam claras, consistentes e orientadas ao usuário. Sua expertise não só facilita a implementação da voz e do tom da marca nos produtos, como também nos componentes do design system. Podemos dizer que são os profissionais que garantem que a comunicação seja absorvida de forma clara, acessível e inclusiva pela pessoa usuária.

3. Desenvolvedores front-end

É impossível ter design systems sem desenvolvedores front-end. São os profissionais fundamentais para tornar realidade tudo o que é planejado para ser utilizado nos produtos da empresa.

Esses profissionais são encarregados de transformar elementos e características visuais em design tokens, assim como as especificações de componentes em código funcional e responsivo. Assim como os designers, eles asseguram que cada interface construída no guia de estilo seja reutilizável, acessível e compatível com diferentes sistemas, navegadores e dispositivos.

Também são responsáveis por garantir bom desempenho em termos de tecnologia, evitando atrasos de carregamento, resolvendo problemas visuais, *bugs* e outras inconsistências que possam surgir.

Capítulo 10
Política de versionamento

Sabe aqueles aplicativos do seu celular ou softwares do seu computador que, de tempos em tempos, pedem atualização para trazer novos recursos, corrigir bugs ou fazer pequenos ajustes? Esse conceito precisa estar presente no design system, que será constantemente evoluído em diferentes aspectos: tecnologia, visual, interação e acessibilidade.

Toda atualização, por menor que seja, de alguma forma impacta as pessoas que estão construindo produtos e que consomem o design system. Uma boa política de versionamento pode ajudá-las a entender o momento atual do produto, as novidades, os problemas resolvidos e, o mais importante, a manter o design system sempre atualizado.

10.1 Versionando seu design system

Em primeiro lugar, é importante estabelecer uma convenção clara de nomenclatura para as versões do design system. A convenção mais comum é a utilização de números em uma certa ordem cronológica, que indica as alterações que foram ocorrendo no produto.

Um ótimo e comum exemplo que você pode seguir é o formato *Semantic Versioning* (https://semver.org/), ou apenas SemVer, representado pelo padrão de "Major.Minor.Patch", que demonstra de forma clara o significado de cada atualização de um produto:

- **Major**: é a indicação da versão principal, representando o estado atual do produto e identificando mudanças significativas e impactantes nos produtos que os utilizam.

- **Minor**: é o número da versão secundária, que representa adições ou atualizações de elementos e componentes.
- **Patch**: é a versão com menor nível de relevância, representando correções de bugs ou outros problemas que possam ter ocorrido.

Nesse cenário, a lógica é que um design system inicie na versão 1.0.0, indicando que ele está na sua fase inicial, ainda sem nenhuma atualização. Conforme as necessidades exigirem evoluções do produto, sairemos dessa versão para a 1.1.0, representando, por exemplo, a inclusão de novos componentes para cenários de formulário, como "Switch" e "Range slider". Se, após a atualização, for identificado algum bug, seja nesses novos componentes ou em outros já lançados na versão inicial, teremos a versão 1.1.1.

```
design-system 1.1.1
├─ styles
└─ components
   ├─ button
   ├─ checkbox
   ├─ radio button
   ├─ range slider
   ├─ select
   ├─ switch
   ├─ text field
   └─ ...
```

É uma demanda que exige muito mais iniciativa por parte das pessoas responsáveis pela tecnologia do design system, já que a atualização é muito mais presente no código do que no design. Contudo, isso não significa que designers e outros profissionais não possam se envolver ou que não devam conhecer esse funcionamento. Afinal, é um trabalho colaborativo, e todos precisam estar a par do estado atual do produto.

Outro aspecto importante da política de versionamento é a definição clara do processo de revisão e aprovação para o envio de novas atualizações. Tudo o que for lançado precisa ser devidamente

testado em todos os aspectos possíveis, evitando publicações que possuam algum tipo de inconsistência.

Para ter um controle maior sobre como e quando essas atualizações devem ser lançadas, torna-se interessante ter pessoas responsáveis pela gestão e liberação. Podem ser a própria liderança da equipe ou, pelo menos, um responsável do time de design e outro de tecnologia, que conheçam o estado atual dos demais produtos e possam indicar quando uma nova atualização deve ser enviada.

10.2 Versionando componentes

Versionar um produto é o cenário mais comum, não só do ponto de vista do design system, mas de qualquer outro, seja software, aplicativo ou até mesmo jogo. No entanto, esse não é o único modelo possível. Também existe a possibilidade de versionar cada componente individualmente, permitindo o uso simultâneo de diferentes versões.

Por exemplo, em um cenário de design de formulário, seria totalmente viável utilizar um *text field* de versão 2.1.0 junto de um *select* de versão 1.3.1.

```
design-system
├─ styles
└─ components
   ├─ button       2.0.0
   ├─ checkbox     2.1.0
   ├─ radio button 2.1.1
   ├─ range slider 1.0.1
   ├─ select       1.3.1
   ├─ switch       1.2.1
   ├─ text field   2.1.0
   └─ ...
```

Esse modelo é interessante para manter atualizações e históricos específicos, pois permite o envio de novidades ou a resolução de *bugs* sem a necessidade de atualizar o design system inteiro. Isso pode tornar o processo mais rápido para modificações em sites ou aplicativos que utilizam a biblioteca, já que as únicas revisões necessárias serão em alguns componentes.

CAPÍTULO 11
Princípios de design

Se você é uma pessoa curiosa, então eu imagino que você já tenha explorado algumas documentações de design systems, não só para tomar como referência, mas também para entender como a empresa em questão produz seus produtos digitais.

Uma das seções que você pode encontrar em alguns design systems é chamada "Princípios". Normalmente, ela está localizada na parte de *"Getting started"*, antes de se aprofundar nos estilos e componentes. O objetivo dessa seção é descrever os princípios que orientam cada interface apresentada no documento.

Princípios são diretrizes fundamentais que orientam as decisões de design e desenvolvimento dentro de um design system. São como pilares conceituais que ajudam a manter consistência, qualidade e alinhamento entre as equipes, mesmo em cenários complexos ou com múltiplos produtos. Ao contrário de regras rígidas, os princípios oferecem uma base para avaliar escolhas e priorizar soluções, refletindo os valores da marca e o propósito das interfaces construídas.

Apesar de ser comum em design systems de grandes marcas e considerado uma boa prática, é importante destacar que não é um pré-requisito. A ausência de princípios não gera problemas de uso nem impactos negativos. No entanto, incluir essa seção pode ser vantajoso, pois orienta as decisões de design e ajuda a manter a coesão dos produtos.

Concordamos que elementos e componentes — tópicos que discutiremos nos próximos capítulos — têm uma contribuição direta e essencial para a padronização. No entanto, as pessoas pensam de formas diferentes e, quando dois ou mais designers utilizam o design

system, a probabilidade de surgirem páginas com decisões divergentes é alta.

Por outro lado, se estabelecermos regras claras para o uso correto e compartilhado de tudo que o design system oferece, as chances de ocorrerem desconexões diminuem significativamente.

Elaborando os princípios de design

A primeira coisa que você precisa entender é que os princípios não pertencem ao design system, mas sim ao design em si. Quando aplicados, todas as decisões de design devem se basear neles. Vale lembrar de que, muitas vezes, a empresa já possui princípios definidos antes mesmo de existir um design system, seja pelo desenvolvimento da marca ou por alguma necessidade interna.

Agora, sobre a forma de definir os princípios de design, existem inúmeras abordagens, uma mais interessante que a outra. Vou apresentar um modelo extremamente interativo, no formato de workshop, proposto pelo especialista em conteúdo de marca Oskar Duberg.

Para se preparar para a dinâmica, separe alguns princípios para fomentar as discussões. Você pode usar sites como **Design Principles** (https://principles.design) ou **Design Principles FTW** (https://www.designprinciplesftw.com) para encontrar referências de princípios reais que as empresas utilizam.

O primeiro passo é convidar algumas pessoas da empresa para participarem do workshop, o que não é uma tarefa difícil. Contudo, é importante lembrar um detalhe: não é porque existe “design” no nome que apenas designers devem participar. Nem todas as empresas possuem equipes de design grandes. Por isso, o ideal é convidar pessoas que estejam atuando de forma mais ativa nos produtos.

Para dinâmicas feitas de forma online, utilize ferramentas de *whiteboard*, como **Miro** (https://miro.com/) ou **FigJam**

(https://www.figma.com/figjam/). Caso você faça presencialmente, imprima pequenos cartões para que possam ser distribuídos entre os grupos.

A dinâmica funciona de forma bem simples. Divida os participantes em grupos, considerando que não deve haver grupos com mais de seis pessoas, pois é importante que todos compartilhem suas opiniões. Em seguida, separe a mesma quantidade de princípios para cada um dos grupos, e peça que descartem somente aqueles que não acreditarem se adequar à empresa.

Nesse momento, haverá muita discussão e troca de ideias, e *insights* podem acabar surgindo, por isso você pode deixar livre a possibilidade de adicionar outros princípios que não foram listados. O ideal é que cada grupo não tenha mais do que dez princípios selecionados.

No fim da dinâmica, reúna todos os princípios e agrupe todos aqueles que se repetem, apontando quais foram os mais populares.

É provável que você acabe com mais de dez princípios, incluindo tanto os repetidos quanto os inéditos. Analise cada um individualmente com os participantes, pois podem surgir opiniões diferentes. Em seguida, realize uma votação, dando três pontos para cada participante escolher os princípios que, na opinião deles, melhor representam o perfil da empresa.

Após a votação, descarte todos os princípios que não receberam votos. Para os que foram selecionados, converse com os participantes para entender quais mudanças seriam necessárias na empresa para que cada princípio pudesse ser adotado com sucesso, avaliando também o custo e o esforço envolvidos.

Analisando alguns princípios

Para variar um pouco, vamos explorar os princípios de outro design system que ainda não mencionei neste livro: o **Fluent design**

system (https://fluent2.microsoft.design), em sua segunda versão, criado pela Microsoft.

O Fluent Design System é baseado em quatro princípios que descrevem como deve ser a experiência das pessoas usuárias. O diferencial, em comparação com outros design systems, é que cada princípio responde a duas perguntas: o que o torna funcional e o que o torna emocional.

O primeiro princípio, chamado de **Natural em todas as plataformas**, enfatiza a importância de proporcionar uma experiência familiar e intuitiva em todos os sistemas. Isso significa que os usuários devem se sentir confortáveis e saber intuitivamente como interagir com a aplicação, seja no Windows ou em outros sistemas, como Android, iOS ou macOS.

O segundo princípio, **Construído para foco**, destaca a importância de criar experiências que permitam aos usuários manterem a atenção e permanecerem no fluxo. Isso significa que as interfaces devem ser simples e contínuas, incentivando a ação e facilitando o progresso do usuário em suas tarefas. Uma experiência funcional sob esse princípio é aquela em que a tecnologia se comunica de forma eficaz e permite que os usuários realizem suas ações sem obstáculos. Para isso, é fundamental evitar elementos desnecessários que possam causar distrações ou confusão, garantindo uma navegação e interação fluida.

Já o terceiro, **Um por todos e todos por um**, retrata a importância de criar experiências inclusivas que considerem e reflitam uma variedade de perspectivas e habilidades. O processo de design deve envolver pessoas com diferentes origens e pontos de vista desde o início, contribuindo para soluções melhores e mais abrangentes. Ao incluir uma ampla gama de pessoas, o design se torna mais criativo e eficaz, abrindo novas possibilidades e ajudando a superar desafios de forma inovadora.

Por fim, mas não menos importante, o quarto princípio do Fluent é chamado de **Inconfundivelmente Microsoft**. Ele garante que o usuário não tenha dúvidas sobre a quem pertence a aplicação com a qual está interagindo. Isso significa que cada produto, serviço ou interação deve refletir a identidade visual e os valores da marca, desde as cores e os ícones até o tom de voz e a personalidade.

Ao focar em experiências exclusivas que conectam os produtos da Microsoft, é possível aumentar o reconhecimento da marca e a familiaridade dos usuários com seus produtos.

Parte 4 - Estilos e design tokens

A identidade visual de um design system é construída a partir de elementos como cores, tipografia, espaçamentos, sombras e bordas. Esses fundamentos garantem a consistência entre diferentes produtos e ajudam a manter a experiência do usuário alinhada à marca. Quando bem estruturados, tornam a aplicação do design mais eficiente e reduzem divergências visuais.

Para organizar esses elementos de forma escalável, são utilizados design tokens, que transformam valores visuais em variáveis reutilizáveis no código. Com isso, ajustes podem ser feitos de maneira centralizada, garantindo que qualquer atualização se propague automaticamente em todos os produtos que dependem do sistema.

Nesta quarta parte do livro, você lerá sobre:

- A importância da acessibilidade durante a interação.
- A seleção e aplicação de cores no design.
- Princípios de tipografia e espaçamento para uma boa legibilidade.
- O conceito de design tokens e sua aplicação nos estilos.

Capítulo 12
Foundation

Se você já explorou algumas documentações de design system populares, provavelmente se deparou com uma seção chamada **foundation**, que costuma ser uma das primeiras apresentadas e, não por acaso, é também uma das mais importantes, pois reúne os elementos básicos e essenciais de praticamente todas as páginas de um produto digital, como cores, sombras e tipografia.

O termo "foundation" é utilizado para se referir à fundação sólida e estruturada de elementos essenciais, que servem como alicerce para a construção de uma interface consistente e coesa. Isso transmite a ideia de uma base sobre a qual o sistema é construído, fornecendo as diretrizes e os elementos fundamentais para a criação de produtos digitais de qualidade.

Sempre que explico sobre esse assunto, costumo comparar *foundations* com pilares de um edifício, que servem como base fundamental para a sua sustentação. Não é diferente com os elementos básicos: basta pensar nos sites e aplicativos que você acessou hoje para perceber isso. Imagine a interface mais simples possível, como um botão — que nada mais é do que um retângulo com uma cor de fundo e um texto indicando a ação. Contudo, se não considerarmos as paletas de cores e os detalhes tipográficos, como fonte, tamanho do texto e até mesmo altura de linha, o resultado seria inexistente. Se esse cenário fosse o de um edifício, ele já teria desmoronado há muito tempo.

12.1 O Atomic Design e a sua aplicação hoje em dia

Para pensar na estrutura de uma página, é comum que as pessoas recorram a metodologias e processos, principalmente quando são iniciantes no assunto. A mais famosa delas, inclusive citada no primeiro capítulo do livro, é o *Atomic Design*, de Brad Frost.

Se seguirmos o conceito da metodologia, entendemos que os elementos mais simples e estruturais de uma interface se tratam dos átomos. Mas a pergunta de 1 milhão de reais: isso tudo é realmente funcional?

Os átomos, caso você não saiba, são considerados a menor unidade funcional da matéria, e estão presentes em literalmente qualquer

material ou objeto, seja ele físico, líquido ou gasoso, representando a construção básica de tudo ao nosso redor. Ao levar isso para o ambiente digital, começamos a pensar em quais são os elementos mais simples que, quando juntos, formam corpos cada vez maiores.

De acordo com o autor da metodologia, os átomos correspondem a elementos básicos do HTML, como botões e campos de formulário, itens com os quais interagimos constantemente. De modo geral, eles podem ser utilizados isoladamente sem comprometer a composição de uma interface. Contudo, quando combinados, passam a formar estruturas mais complexas, sendo importante observar que não podem ser divididos sem perder sua funcionalidade.

Isso se torna funcional do ponto de vista da criação de componentes, e é inegável que desde a criação da metodologia muitas pessoas e empresas se beneficiaram desse processo. Além disso, muitas propostas de melhorias surgiram ao longo dos anos, considerando a evolução dos produtos e o surgimento de novos conceitos, que despertaram até mesmo a atenção de Brad Frost, que em 2019 publicou em seu site pessoal o artigo *"Extending Atomic Design"* (Estendendo o Atomic Design).

A motivação do autor ao publicar essa metodologia foi incluir uma nova etapa anterior aos átomos, inspirada em partículas subatômicas, como os íons. O objetivo dessa etapa é estruturar adequadamente os componentes que surgem na fase dos átomos, utilizando, nesse caso, os **design tokens**.

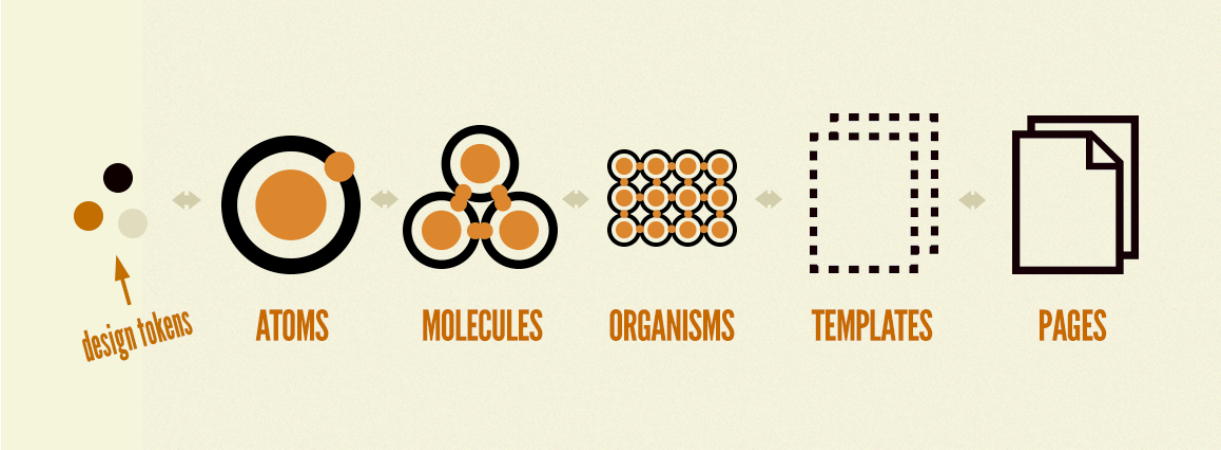


Figura 12.1: A imagem mostra as etapas do Atomic Design, com uma nova primeira etapa na frente, chamada "design tokens". Nas restantes, estão as etapas "átomos", "moléculas", "organismos", "templates" e "páginas".

Oficialmente, a metodologia não evoluiu para incluir essa nova primeira etapa. Sabemos que, como qualquer processo de design, ela pode ser facilmente adaptada por profissionais para atender a diferentes necessidades, afinal, solucionar problemas faz parte do nosso trabalho. Para o autor da metodologia, não há necessidade de modificar as etapas, justamente porque tudo está em constante mudança, e adaptar o processo a cada novidade não seria o mais adequado.

Particularmente, não vejo como começar o guia de estilo de um design system iniciando diretamente pelos átomos. É como se estivéssemos pulando tudo o que definirá a identidade do nosso produto. Botões não só podem, como devem ser divididos, ao ponto de pensarmos na cor, borda, espaçamento, tipografia e qualquer outro elemento que possa ser encaixado.

Até o final desta parte do livro, falaremos sobre design tokens, possivelmente um dos assuntos que mais exigirá sua atenção. Não vou enganar você: à primeira vista, trabalhar com tokens pode parecer complicado, mas a prática torna tudo cada vez mais simples. Para chegarmos lá, primeiro precisamos compreender cada detalhe básico que torna uma interface única e eficiente.

CAPÍTULO 13
Acessibilidade

Antes de nos aprofundarmos no assunto de acessibilidade, é fundamental compreender um detalhe importante: acessibilidade não se destina apenas a pessoas com alguma condição ou deficiência, mas a todos, sem exceção. Quando um produto está disponível na internet, espera-se que ele possa ser utilizado sem dificuldades; caso contrário, a experiência do usuário será certamente negativa.

Quando falamos em acessibilidade em um design system, queremos dizer que tornaremos nossas interfaces fáceis e utilizáveis por qualquer pessoa. Não é uma tarefa simples, já que o tema é denso, mas considero essa etapa vital para qualquer produto digital. Apesar da complexidade, não estamos perdidos em relação ao conteúdo, pois contamos com as **WCAG** como guia.

A melhor referência para iniciarmos os trabalhos de acessibilidade está na *Web Content Accessibility Guidelines* (Diretrizes de Acessibilidade para Conteúdo da Web), mais conhecida como WCAG. Esse guia foi criado pela W3C (*World Wide Web Consortium*) e estabelece uma série de diretrizes e critérios de sucesso para criar páginas web acessíveis. Cada critério possui um dos três níveis definidos, que indicam o grau de acessibilidade alcançado:

- O nível mais baixo é identificado como **A**;
- O nível intermediário é identificado como **AA**, que engloba também os critérios do nível anterior;
- O nível mais alto é identificado como **AAA**, que engloba também os critérios dos níveis anteriores.

Esse é um assunto que exige calma e atenção especial para estudo. A explicação que você acabou de ler é só uma pontinha do *iceberg* e, como a WCAG é um guia longo (até a data de publicação deste

livro, não há uma versão oficial em português), eu recomendo que você acesse o site *Guia WCAG* (https://guia-wcag.com/), criado pelo Marcelo Sales, fundador e CEO do *Tudo é Acessibilidade*, também uma das referências em acessibilidade no Brasil.

Existem diversas formas de começar a trabalhar a acessibilidade nas interfaces do seu design system, ou até mesmo de aprimorar o que já foi implementado. A seguir, apresento as principais práticas que tornarão os produtos que consomem o seu design system acessíveis para grande parte do público.

13.1 Contraste de cores

O contraste de cores se refere à diferença perceptível entre as cores do plano superior e o plano de fundo, sendo algo essencial para garantir que o conteúdo seja legível e compreensível para todas as pessoas, incluindo aquelas com deficiências visuais ou dificuldades de leitura. Normalmente, o contraste de cores costuma estar envolvido diretamente com a leitura, levando em conta a cor do texto em relação à cor do fundo.

Para validar se um contraste está realmente eficiente, ou se ainda há possibilidade de dificuldades de leitura ou visualização, basta acessar plataformas que fazem a análise de modo automático, inserindo somente as cores de primeiro e segundo plano.

Como sugestão, recomendo o site *Colour Contrast Checker* (https://colourcontrast.cc/), que oferece uma visão completa do contraste e ainda traz uma área com texto de exemplo para verificar a aplicação na prática. Seguindo os padrões da WCAG, essa ferramenta possibilita não apenas identificar em qual dos três níveis de acessibilidade as cores se enquadram, mas também conhecer a **relação de contraste** entre elas.

Contraste mínimo

O critério de sucesso para que um contraste seja considerado minimamente acessível é que ele alcance o nível AA, o que também pode ser identificado pelo valor de relação de contraste, que deve ser pelo menos 4.5:1. Contudo, existem algumas exceções que autorizam que esse valor seja menor, como é o caso de textos em tamanhos grandes (24 pixels, 19 pixels em negrito ou fontes equivalentes a caracteres em chinês, japonês ou coreano), que devem apresentar no mínimo uma relação de contraste de 3:1.

Contraste alto

O critério de sucesso para que o contraste entre as cores atinja o nível mais alto de acessibilidade é que ele esteja no nível AAA, com relação de contraste a partir de 7:1. Semelhante ao contraste mínimo, também há a exceção para textos com tamanhos grandes, seguindo os mesmos detalhes de tamanho e espessura, com a diferença de que a relação de contraste mínima deve ser de 4.5:1.

Contraste não textual

Como eu disse anteriormente, o cenário mais comum da relação de contraste é entre a cor de um texto, que está no primeiro plano, com um plano de fundo. Mas claro que esse não é o único elemento que impacta na acessibilidade com as cores. Contrastes não textuais possuem uma relação de contraste mínima de 3:1, permanecendo no nível AA na maioria das circunstâncias.

Os primeiros itens, citados pela WCAG, são os **componentes de UI**, que possuem informações visuais que necessitam ser bem visualizadas para que as pessoas possam identificar o estado correto. Outro desses elementos são **partes de gráficos**, que precisam ser acessíveis para conseguir diferenciar os itens e entender o conteúdo.

Exceções gerais

Você deve ter notado que alguns dos itens anteriores possuem algumas "brechas" específicas liberadas pela própria WCAG, que

permitem ter uma relação de contraste menor, mas mantendo o nível de acessibilidade. Ainda assim, existem mais dois itens que servem como exceções gerais para todos os casos de acessibilidade, que não possuem requisitos de contraste.

O primeiro caso diz respeito a **textos para estados inativos ou decorativos**, que não possuem funcionalidade nem fornecem informações — como campos de texto desabilitados. Estar inativo não significa apenas a impossibilidade de interação, mas também que o elemento perde o foco, já que não precisa estar disponível para uso.

O segundo e último item referem-se a **textos que fazem parte de um logotipo**, já que a marca possui padrões que devem obrigatoriamente ser respeitados, não cabendo modificações nesse elemento. Ainda assim, é fundamental usar o bom senso: se o logotipo de uma marca for vermelho, por exemplo, não deve ser aplicado sobre um fundo laranja — isso seria um verdadeiro crime contra a acessibilidade.

13.2 Legibilidade de texto

A legibilidade é um conceito essencial que trata da capacidade de tornar um texto claro e fácil de ser lido e compreendido. Ter uma boa legibilidade é fundamental para garantir uma experiência positiva para todas as pessoas usuárias enquanto estão lendo alguma coisa presente em um site, aplicativo ou qualquer outro produto digital, independentemente de suas habilidades visuais ou até mesmo do contexto de leitura.

Apesar de não ser a única, a escrita via texto é uma das maiores formas de passar informações ao acessar, por exemplo, um site. Na internet, se queremos buscar alguma coisa, normalmente digitamos

isso no Google, e vamos lendo e procurando até encontrarmos o que precisamos.

Tenha em mente que, sempre que uma pessoa precisar forçar a vista para ler algo, significa que a interface está falhando com ela. Por essa razão, essa diretriz estabelece alguns requisitos que precisam da sua atenção para que a leitura seja algo fácil, sendo algumas delas:

- Escolha da fonte ideal;
- Tamanho do texto;
- Altura entre as linhas;
- Espaçamento entre as letras (ou simplesmente "*tracking*") e também entre as palavras.

Escolha da fonte ideal

É bastante comum que a fonte escolhida para o design system — usada em todos os produtos que o consomem — seja a mesma do logotipo da empresa ou aquela indicada no manual da marca. Isso não está errado; na verdade, pode ser bastante positivo, já que a tipografia também é uma forma de transmitir a mensagem da marca. No entanto, a fonte utilizada pela marca nem sempre é a escolha ideal para interfaces digitais, seja por questões de estilo ou por outros fatores. Quer um exemplo disso?

O logotipo do Instagram, que dispensa apresentações, utiliza a fonte Billabong, que possui um estilo cursivo (com letras emendadas). Para quem já usou alguma vez o aplicativo, sabe que não há nenhum outro texto entre as páginas que utilize essa mesma fonte ou letra cursiva. Na realidade, a plataforma nem possui uma fonte padrão para ser exibida: ela varia conforme o sistema operacional que você está acessando, mas a empresa teve a ciência de que, apesar da estética visual ser agradável, usar o mesmo estilo de texto do logotipo para todas as informações no aplicativo poderia ser algo prejudicial para a leitura, considerando que letras cursivas,

principalmente em tamanhos pequenos, podem ser mais difíceis de serem lidas.

Figura 13.1: Logotipo do Instagram, formado por uma fonte cursiva em que todas as letras estão emendadas, com cor cinza escuro.

Fontes sem serifa são frequentemente consideradas mais adequadas para uso em telas digitais, pois apresentam aparência mais limpa, objetiva e legível em tamanhos menores. Geralmente, possuem linhas retas e não contam com os traços decorativos nas extremidades das letras (as chamadas serifas), o que facilita a leitura. Bons exemplos de fontes amplamente utilizadas em diferentes produtos são Inter, Open Sans e Roboto — todas gratuitas.

Entretanto, a legibilidade não depende apenas de estilo. A quantidade e a variedade de caracteres disponíveis em uma fonte são fundamentais, especialmente considerando os idiomas nos quais a marca transmitirá informações.

Só no português, por exemplo, há diversos caracteres acentuados, e a ausência desse suporte já representa uma falha grave: além de prejudicar a imagem da marca, pode dificultar a compreensão para pessoas que utilizam leitores de tela.

Não é necessário escolher uma fonte com suporte para centenas de idiomas e regiões, como a Roboto, que abrange mais de 300 idiomas, mas sim uma que atenda plenamente às suas necessidades. Portanto, antes de bater o martelo, valide os

diferentes cenários: números, acentos e até mesmo caracteres especiais.

13.3 Navegação por teclado

Sabemos que o mouse é um dispositivo extremamente comum para usar um computador, já que é o que permite clicarmos e navegarmos com praticidade por sistemas e sites. Mas convenhamos, para determinadas ações, o teclado se torna o meio mais prático de todos. Por exemplo, ao entrar em uma página de login, é comum usar o mouse para clicar no campo de e-mail, e em vez de repetir esse mesmo movimento para a senha, pressionar a tecla “Tab” do teclado para trocar de um campo para o outro.

O exemplo dado pode parecer apenas uma questão de praticidade para pessoas usuárias experientes — e de fato é —, mas o ponto principal está em considerar como pessoas com algum tipo de deficiência visual poderão utilizar produtos digitais, em computadores ou dispositivos móveis, sem barreiras ou dificuldades de usabilidade.

Caso você não possua alguma deficiência visual, como cegueira ou baixa visão, e não saiba como essas pessoas utilizam computadores, posso garantir que não é muito diferente da forma como pessoas que enxergam bem usam. Talvez, o maior diferencial seja o uso dispensável do mouse, já que é um objeto que de fato depende da visão para se controlar o cursor na interface. Nesse caso, o teclado substitui completamente o mouse. Embora não ofereça a mesma forma de interação, viabiliza a navegação por meio de atalhos como “Tab” para avançar entre itens, “Shift + Tab” para retornar e “Enter” para executar uma ação. Quando aliado a softwares de leitura de tela, torna possível utilizar um computador sem grandes dificuldades — ou, pelo menos, é assim que deveria ser.

Eu adoraria finalizar essa parte do capítulo dizendo que, ao utilizar apenas essas e outras teclas de atalho, o mouse se tornaria obsoleto. Infelizmente, não é bem assim: o teclado não é inteligente o suficiente para saber a ordem correta das informações e ações, permitindo que pessoas que realmente dependem dele naveguem com facilidade, saibam onde estão, para onde podem ir e encontrem o que buscam. Por isso, é essencial pensar na estrutura dos componentes e das páginas, bem como nas diferenças de interação entre computadores e *smartphones*.

Utilize a estrutura semântica em HTML

Podemos considerar que o ponto de partida para garantir a acessibilidade na navegação por teclado em sites é o uso adequado da estrutura em HTML. Caso você não seja programador(a) e não tenha familiaridade com essa linguagem, eu explico: o HTML não é uma linguagem de programação, mas sim de marcação de texto. Isso significa que ele facilita identificar e diferenciar diversos tipos de informação em uma página, como textos, ações, blocos e até seções inteiras de conteúdo.

Uma das formas de começar a organizar uma página é utilizando seções. É comum recorrer à tag `<div>` sempre que se deseja criar um novo bloco na construção de um site. No entanto, o que muitas pessoas não sabem é que esses blocos não são percebidos pelo teclado, como se simplesmente não existissem na navegação.

Para contornar isso, há uma solução simples: adicionar o atributo `tabindex`. Com ele, a navegação por teclado consegue identificar e focar nessa área, sendo possível ainda definir a ordem de leitura por meio de um número atribuído ao atributo. Ou seja:

- 0 para definir em ordem natural;
- Números positivos (1, 2, 3…) para definir a posição específica de foco, considerando os elementos da página;
- -1 para que o elemento não seja identificado usando o atalho "Tab".

Dessa forma, o resultado final do código é representado da seguinte maneira: `<div tabindex=”0”> … </div>`

Outra solução, e até mesmo mais recomendada para esse caso, é utilizar as outras tags que o HTML oferece para criar e diferenciar seções, como `<header>`, `<main>`, `<nav>`, `<section>`, `<article>`, entre outras.

Para otimizar a qualidade das informações textuais apresentadas em uma página, é importante diferenciar corretamente os elementos. Os títulos devem ser marcados com as tags que vão de `<h1>` a `<h6>`, enquanto os parágrafos devem utilizar a tag `<p>`. Da mesma forma, listas também precisam ser categorizadas de maneira adequada: listas ordenadas com `<ol>` e não ordenadas com `<ul>`.

Pular para o conteúdo principal

Esse é um item que quem utiliza o mouse dificilmente percebe, mas que está presente no topo de muitos sites. Trata-se de um link que, em alguns cenários, pode ser focado logo no início pelo leitor de tela. Com ele, a pessoa usuária pode interagir pelo teclado e ir diretamente ao conteúdo principal, se assim desejar; caso contrário, pode continuar navegando pelos elementos anteriores da página.

O próprio Google possui esse recurso em sua página de busca, embora ele fique oculto até receber foco. Para visualizar esse simples link, basta realizar qualquer pesquisa e pressionar a tecla “Tab” uma única vez. Logo abaixo do logotipo do Google, aparecerá o link “Pular para o conteúdo principal”.

Se implementar esse recurso já é simples, entender a sua motivação é ainda mais. Pessoas que enxergam normalmente podem, com facilidade, direcionar o olhar para a informação desejada — como no caso do Google, ao visualizar os links retornados na busca. Se isso é natural para elas, por que pessoas com deficiência visual não deveriam ter a mesma possibilidade de acesso?

Cores para identificar ações

Já comentamos sobre acessibilidade no uso das cores e a forma como os contrastes podem acabar dificultando a usabilidade e o entendimento se não forem bem trabalhados. Mas esse não é o único aspecto em que as cores desempenham um papel importante. Elas também são fundamentais para indicar quando uma ação está em foco, sinalizando ao usuário que é possível interagir com aquele elemento.

A navegação por "Tabs" permite que as pessoas usuárias avancem em campos, botões, links e outros elementos clicáveis de uma página. Por mais que um recurso visual não faça diferença para pessoas cegas, é importante lembrar que existem outras condições visuais que dificultam a percepção. Nesses casos, o uso de cores para destacar o foco representa um ganho significativo, ajudando a diferenciar o que é navegável do que não é.

Você pode fazer esse teste prático no próprio computador — e eu recomendo fortemente, sobretudo se você não apresenta dificuldades de visão. Para o seu teclado, não é necessário instalar nenhum recurso extra: ele já oferece as teclas necessárias para navegar normalmente, seja em um software ou em um site. O ponto essencial dessa validação é utilizar um programa de leitura de tela, simulando o cenário de uma pessoa que não enxerga ou não consegue distinguir as informações. Nesse caso, a audição se torna o sentido principal para realizar essa "leitura".

Dessa forma, conforme você for pressionando teclas "Tab", "Espaço", "Enter" ou mesmo as teclas de setas direcionais, o software de leitura deverá retornar a mensagem correspondente — seja uma ação clicável ou apenas uma informação. Caso contrário, a navegação se torna confusa e a pessoa usuária terá dificuldade em se situar na página.

À medida que o design system passa a incorporar recursos de acessibilidade, é fundamental disseminar essa prática entre as

demais equipes. Mais do que aplicar, elas precisam também validar esses recursos em seus produtos — não apenas de forma interna, mas sobretudo junto a usuários da plataforma que tenham algum tipo de deficiência.

13.4 Textos alternativos para elementos não textuais

Assim como os textos, é muito comum que produtos digitais utilizem imagens, ilustrações, ícones e outras mídias essencialmente visuais. Pessoas com deficiência visual podem não enxergar esses elementos da mesma forma que quem vê normalmente, mas isso não significa que não possam compreender o que está sendo apresentado na página.

Para contornar essa questão, devemos conhecer e utilizar os **alternative texts**, ou apenas *alt texts*, como são comumente chamados. Esse é um atributo extremamente simples e fundamental do HTML, que permite adicionar uma descrição textual às imagens. Ele é útil não apenas quando a imagem não pode ser exibida — por exemplo, em conexões instáveis ou quando o arquivo foi removido do site —, mas também para pessoas que não conseguem enxergá-la total ou parcialmente.

No caso dessas pessoas, que normalmente utilizam softwares de leitura de tela, é importante que a descrição seja clara o suficiente para proporcionar uma compreensão equivalente à de quem pode enxergar a imagem, especialmente quando ela é relevante para o contexto da página.

Vamos supor que uma pessoa cega acesse um artigo sobre tucanos para entender melhor sobre essa ave. Os tucanos são aves bem peculiares não só pela questão da região em que costumam viver, mas também pela sua característica física. É uma ave de porte

médio, de bico longo, grosso e curvado, com cores vivas como laranja, amarelo e vermelho. Dependendo da espécie, há detalhes na cor preta.

Note que, embora a descrição do tucano não seja totalmente detalhada, eu destaquei a principal característica que o diferencia de outras aves. São exatamente esses detalhes que precisamos incluir para fornecer todo o contexto necessário a usuários de leitores de tela.

Posto isso, trago duas ótimas práticas para a produção de textos alternativos que vão ajudar você a elaborar textos da maneira correta:

- Seja descritivo no texto alternativo: descreva o contexto de cenários, pessoas, objetos e tudo o que de fato for relevante para a compreensão da imagem. Esqueça descrições como "imagem de um pássaro", e opte por detalhes maiores, como "a imagem mostra um céu azul, sem nuvens, com um canário de cor amarela voando".
- Seja objetivo ao escrever textos alternativos: eles devem ser descritivos, mas concisos. Apenas em casos específicos, quando for necessário compreender bem o contexto, a descrição pode ser um pouco mais longa do que o usual.

Lembre-se de que não é porque uma pessoa possui limitações que ela deve ter um conteúdo apresentado de modo diferente. Quando falamos de experiência do usuário, a empatia é fundamental, reforçando a importância de incluir a acessibilidade em todos os aspectos do design.

Se ainda não faz parte da sua rotina, sugiro instalar alguma ferramenta de leitura de tela nos seus dispositivos. Isso permitirá não apenas entender como funciona esse tipo de navegação, mas também validar se o seu design system é realmente acessível, além de possibilitar testes diretos com usuários sempre que possível.

13.5 Uso de ARIA Roles

A acessibilidade na web é um desafio significativo, especialmente quando se trata de usuários que dependem de tecnologias assistivas, como leitores de tela para navegar em sites e aplicativos. Muitos elementos de interface complexos, como menus dinâmicos, sliders e modais, podem ser difíceis de entender e utilizar para pessoas com deficiência, devido à falta de informações claras sobre o comportamento e o propósito desses elementos. Isso gera uma experiência frustrante e, frequentemente, inacessível, excluindo um grande número de usuários do acesso completo aos conteúdos e funcionalidades da web.

Para resolver esses problemas, foi criada a especificação ARIA (*Accessible Rich Internet Applications*) pela W3C, que define um conjunto de atributos que podem ser adicionados ao HTML para fornecer informações adicionais sobre os elementos da interface. Esses atributos ajudam tecnologias assistivas a interpretar corretamente o comportamento e o estado dos elementos, permitindo, por exemplo, indicar se um menu está aberto ou fechado, ou identificar corretamente um botão, mesmo que ele não seja implementado como um elemento `<button>`.

Com o uso correto dos atributos ARIA, é possível criar interfaces mais inclusivas, garantindo que todos os usuários, independentemente de suas capacidades, possam interagir de forma eficaz com o conteúdo.

Pensando nisso, trago alguns dos atributos ARIA que você mais poderá utilizar no seu design system:

aria-label

O atributo `aria-label` é usado para fornecer um rótulo acessível a um elemento que não possui texto visível ou cujo texto visível não descreve suficientemente a função do elemento. Por exemplo, em um ícone de busca que é representado apenas por um ícone de

lupa, o aria-label pode ser usado para garantir que leitores de tela leiam a palavra "Buscar".

```
<button aria-label="Buscar">
```

aria-hidden

O atributo `aria-hidden` é usado para esconder elementos de tecnologias assistivas, como leitores de tela. Isso é útil em casos em que elementos são puramente decorativos ou quando você deseja evitar que um elemento seja anunciado por um leitor de tela, como ícones que não têm importância semântica.

Vemos isso com frequência em botões: ao lado esquerdo do texto, há um ícone que ajuda a representar a ação, facilitando seu reconhecimento visual. Contudo, esse tipo de comunicação é mais eficaz para pessoas que enxergam. Para usuários de tecnologias assistivas, uma abordagem direta ao ponto tende a ser mais eficiente e assertiva.

```
<button class="card">
<svg aria-hidden="true">...</svg>
Salvar
</button>
```

aria-expanded

O atributo `aria-expanded` é usado em elementos que controlam a exibição de conteúdo colapsável, como menus *dropdown* ou *accordions*. Ele indica se o conteúdo associado está atualmente expandido ou colapsado:

```
<button aria-expanded="false" aria-controls="section1">
Seção 1
</button>
<div id="section1" hidden>
Conteúdo da Seção 1
</div>
```

aria-live

O atributo `aria-live` é usado para informar às tecnologias assistivas que o conteúdo de um elemento vai mudar e que essas mudanças devem ser anunciadas ao usuário. É extremamente útil em mensagens de *feedback* dinâmicas, como é o caso das notificações.

```
<div aria-live="assertive">
Ocorreu um erro ao salvar seus dados.
</div>
```

aria-modal

O atributo `aria-modal` é utilizado para indicar que o conteúdo de um diálogo deve ser tratado como modal, ou seja, o usuário deve interagir com o diálogo antes de voltar a interagir com o restante da página. Isso é importante para garantir que, ao abrir uma janela na própria página, o foco permaneça dentro desse componente, impedindo o acesso a outros elementos da interface até que o modal seja fechado.

No caso de modais, ainda há mais dois atributos ARIA que auxiliam na acessibilidade, garantindo uma experiência de navegação consistente e previsível para usuários de tecnologias assistivas: o `aria-labelledby`, que identifica o título da janela, e o `aria-describedby`, que descreve o conteúdo presente no modal.

```
<div
role="dialog"
aria-modal="true"
aria-labelledby="dialog-title"
aria-describedby="dialog-description">
<h2 id="dialog-title">Título da Modal</h2>
<p id="dialog-description">Descrição ou conteúdo da modal.</p>
<button>Fechar</button>
</div>
```

Considere que essas são apenas algumas das propriedades que eu julguei importantes para você conhecer, especialmente se tiver pouco contato com o assunto. Recomendo que se aprofunde nesse

recurso e explore as demais propriedades que podem ser aplicadas para tornar seus projetos mais acessíveis.

13.6 Foco visível

Imagine um usuário navegando em um site utilizando apenas o teclado. Ao tentar acessar um formulário e pressionar a tecla "Tab" para passar de um campo para outro, ele não encontra nenhum indicativo visual claro do elemento selecionado. Isso faz com que o usuário fique perdido, sem saber se está no campo correto ou se avançou para a próxima parte da interface.

Essa falta de indicativo não só causa frustração como também torna a navegação ineficiente, especialmente para aqueles que dependem exclusivamente do teclado para interagir com a interface. Felizmente, há uma forma eficaz e fácil de se resolver isso utilizando o foco visível.

O foco visível é um dos vários critérios de sucesso (número 2.4.7) da WCAG, sendo um indicador visual que destaca o elemento selecionado em uma página, como links, botões ou campos de formulário quando navegamos através do teclado. Costumamos ver isso em formato de borda ao redor de um elemento clicável, que inclusive é a recomendação da própria WCAG. Entretanto, há diferentes formas de se criar foco visível em um componente. Contudo, há algumas exigências definidas que merecem atenção.

A primeira exigência é com relação a uma área mínima de trabalho. Caso você decida seguir com o modelo de borda, é importante que ela possua pelo menos 2 pixels de espessura. Se o elemento não for retangular, ele pode seguir seu próprio formato ou criar uma caixa retangular ao redor, delimitando o espaço.

Trabalhar com bordas não é a única forma, apesar de ser a mais comum. É possível customizar o foco em um elemento de modo que

o indicador seja uma mudança de cor, ou até mesmo um aumento no tamanho do próprio elemento. Isso exige que você verifique se a área do indicador de foco atinge, matematicamente, uma área mínima. Recomendo se aprofundar no *critério de sucesso 2.4.13* (https://www.w3.org/WAI/WCAG22/Understanding/focus-appearance.html), chamado *"Aparência do foco"*.

Outro fator diz respeito ao contraste da cor do foco visível. Independentemente do modelo utilizado — seja a borda clássica, uma sombra ou até mesmo um sublinhado —, a relação de contraste com o fundo deve ser de, no mínimo, 3:1, garantindo que o foco não passe despercebido, principalmente por pessoas com deficiência visual.

13.7 Mensagens de erro claras

Imagine o cenário em que um usuário está tentando preencher um formulário em um site de compras, e que um dos campos obrigatórios é deixado em branco por engano. Após clicar no botão "Enviar", o formulário é recarregado sem fornecer qualquer feedback visível ou audível para indicar o problema. O usuário não sabe por que o formulário não foi enviado e pode acabar frustrado, especialmente se tiver alguma deficiência visual ou cognitiva, que dificulta ainda mais a compreensão do que deu errado.

Esse é um cenário claro de como a falta de acessibilidade nas mensagens de erro pode criar barreiras significativas para os usuários. Convenhamos que ninguém gosta de errar, mas isso pode acontecer e, para evitar que seja tão frustrante, a WCAG traz alguns critérios de sucesso que indicam formas de criar comunicações de erro adequadas.

O critério de sucesso 3.3.1, denominado *"Identificação do erro"*, exige que os erros de entrada sejam identificados automaticamente

e que o usuário seja informado da ocorrência de maneira compreensível, tanto em formato visual quanto audível.

A WCAG considera como erros de entrada qualquer informação fornecida pelo usuário que não seja aceita pelo sistema. Isso pode envolver, por exemplo, uma informação omitida em um campo obrigatório, como a senha em um formulário de login, ou um valor fora do formato aceitável, como uma data de nascimento inserida dois anos no futuro.

Para complementar a forma como as mensagens de erro são apresentadas, temos também o critério de sucesso 3.3.3, chamado *"Sugestão de erro"*. Esse item recomenda que, sempre que possível, sejam oferecidas sugestões para corrigir os erros, especialmente em campos que exigem respostas em formatos específicos, como números de telefone ou datas. Isso ajuda os usuários a entender não apenas que um problema ocorreu, mas também como resolvê-lo de maneira rápida e com pouco esforço.

Capítulo 14
Cores

A maior parte dos objetos ao nosso redor possui cor e, por mais que esse aspecto possa estar ligado diretamente a um efeito visual mais bonito e harmônico, seja no mundo real ou no ambiente digital, as cores possuem diversos propósitos diferentes.

Da mesma forma que um semáforo mostra a cor laranja para sinalizar atenção no trânsito, levando a pessoa a preferir frear e esperar em vez de seguir em frente, produtos digitais como sites e aplicativos também possuem vários desses sinais que nos fornecem indicações do que aconteceu ou pode vir a acontecer. Embora a mensagem de texto seja a melhor forma de entender completamente o contexto, ao observar uma cor podemos ter uma noção do que está acontecendo.

Todo guia de estilo de um design system deve contemplar diversas paletas de cores, considerando os diferentes cenários que podem surgir ao longo dos fluxos do produto, incluindo textos, ilustrações e interações com os componentes. No entanto, não basta apenas adicionar muitas cores ao design system sem compreender seu uso específico ou prepará-las corretamente.

De modo geral, não há mistério nas paletas a serem trabalhadas em um design system, mas é importante compreender seus conceitos e o que você pode alcançar com cada uma das cores, além da forma correta de se trabalhar com elas, levando em conta também os critérios de acessibilidade abordados no capítulo anterior.

14.1 Paletas de cores

Cores da marca

As cores da marca desempenham um papel crucial na criação de uma identidade visual forte e memorável. Elas ajudam a diferenciar a marca da concorrência e a construir reconhecimento entre os consumidores conforme eles utilizam os produtos da empresa.

No contexto de um produto digital, as cores de uma marca são documentadas e padronizadas para garantir consistência em todos os produtos e plataformas. Em alguns casos, isso envolve a construção de uma paleta única; em outros, é necessário dividir as cores em primárias e secundárias, podendo até expandi-las. Com uma ou mais paletas de cores da marca, a finalidade sempre será a mesma: indicar ações principais e complementares que podem tanto refletir um desejo quanto representar a identidade da empresa.

Quando falamos apenas da "cor da marca" ou, em alguns casos, da "cor primária", nos referimos à cor mais importante e à mais utilizada dentre as paletas de cores da marca. Seu uso é reservado para os elementos com os quais o usuário mais interage, como botões e links, mas também para indicar algo ativo ou selecionado, como no caso de menus e tabs.

Nos casos em que uma marca possui mais de uma cor na sua identidade, é natural que haja as paletas de cor "secundária" e assim por diante. O uso delas não deve atrapalhar as interfaces que utilizam a cor primária, mas sim complementar. A aplicação dessas cores pode estar presente em componentes cuja interação seja menor, como paginações, gráficos e legendas.

Quanto aos nomes "primário" e "secundário", peço que não se apegue. Essa é apenas uma forma que se tornou muito comum para fazer referência às cores de uma identidade visual e à sua hierarquia dentro dos produtos digitais.

Cores de feedback

As cores de feedback, também chamadas de cores de status ou cores semânticas, são um conjunto de paletas essenciais para fornecer feedback visual imediato. Elas ajudam os usuários a

entenderem rapidamente o que está acontecendo e como devem proceder. A aplicação correta dessas cores melhora significativamente a usabilidade e a experiência, tornando as interações mais intuitivas e eficientes.

Esse grupo de cores de feedback é composto, ao todo, por quatro paletas: *danger*(perigo), *informational* (informativo), *success* (sucesso) e *warning*(aviso), que devem ser cuidadosamente utilizadas ao longo das páginas para evitar que a comunicação seja interpretada de forma equivocada.

Vamos começar falando sobre a paleta de cores **danger**, que é composta por tons de vermelho. Essa paleta é reservada para situações críticas ou erros, que exigem a atenção imediata do usuário. Em uma página, quando um elemento ou mensagem é realçado com a cor vermelha, ele comunica claramente que algo deu errado, ou que há uma ação necessária para evitar problemas futuros.

Existem diversos cenários em que feedbacks em *danger* podem aparecer, como é o caso de formulários. Tenho certeza de que em algum momento você já errou seu e-mail ou senha ao tentar fazer login em um site, e uma mensagem de cor vermelha apareceu logo abaixo do campo, explicando que a informação digitada estava incorreta. Outro momento comum é ao excluir algo, como uma conta de um site, que não reflete erros, mas sim uma ação crítica que pode ser irreversível.

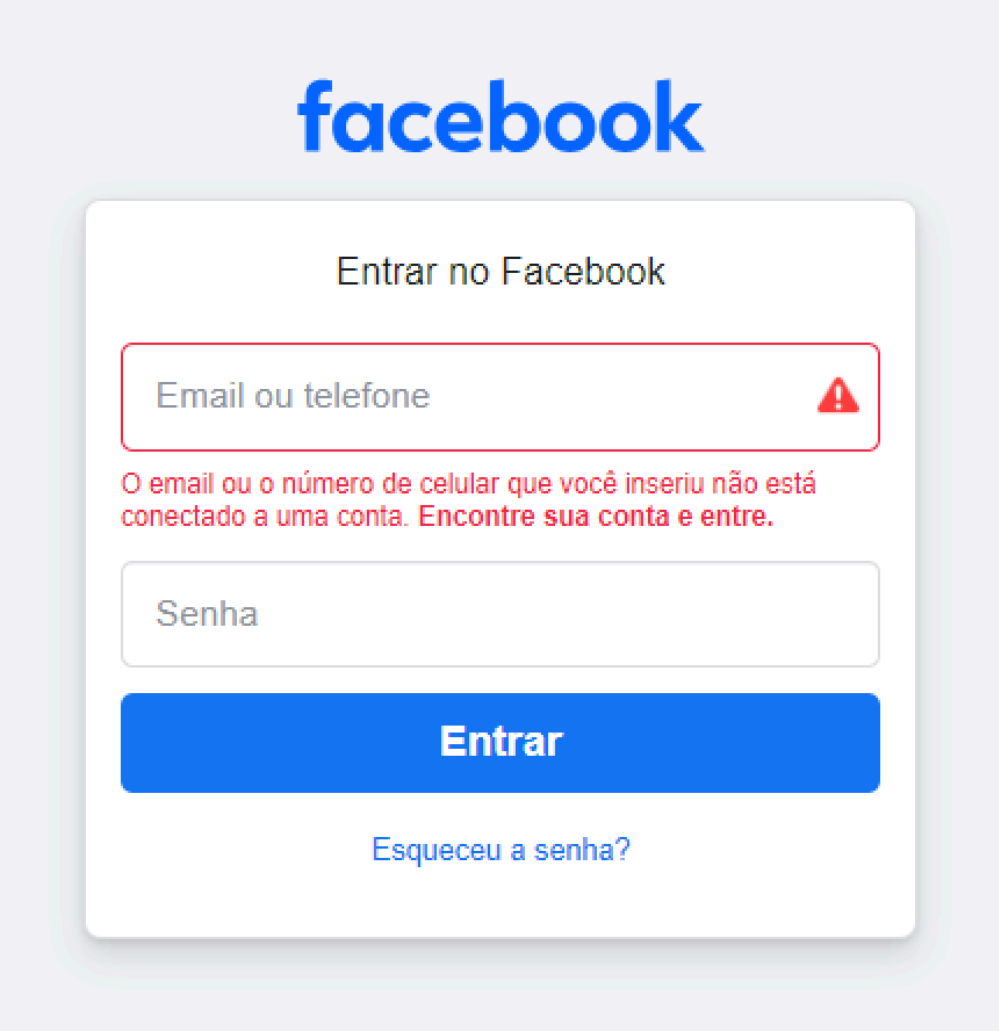


Figura 14.1: Área de login do Facebook. Centralizado no topo está o logotipo do Facebook. Abaixo está um card com os campos de e-mail e senha, o botão "Entrar", e o link "Esqueceu a senha". O campo de e-mail está com borda vermelha, e dentro dele, no lado direito, está um ícone de triângulo com uma exclamação dentro, na cor vermelha, sinalizando alerta. Abaixo do campo está escrito "O e-mail ou o número de celular que você inseriu não está conectado a uma conta. Encontre sua conta e entre".

Se a paleta *danger* tem como objetivo informar erros ou alertas críticos que possam ocorrer no produto, nada mais justo do que termos uma outra paleta que informe o contrário. Para isso, temos a paleta de cores **success**, utilizando tons de verde. Como o próprio nome entrega, serve para indicar ações bem-sucedidas e quaisquer outros estados positivos.

O cenário mais comum para feedbacks de sucesso está na confirmação após a conclusão de alguma tarefa, como um cadastro, edição, confirmação ou qualquer outra ação possível que tenha sido realizada de forma correta e efetiva.

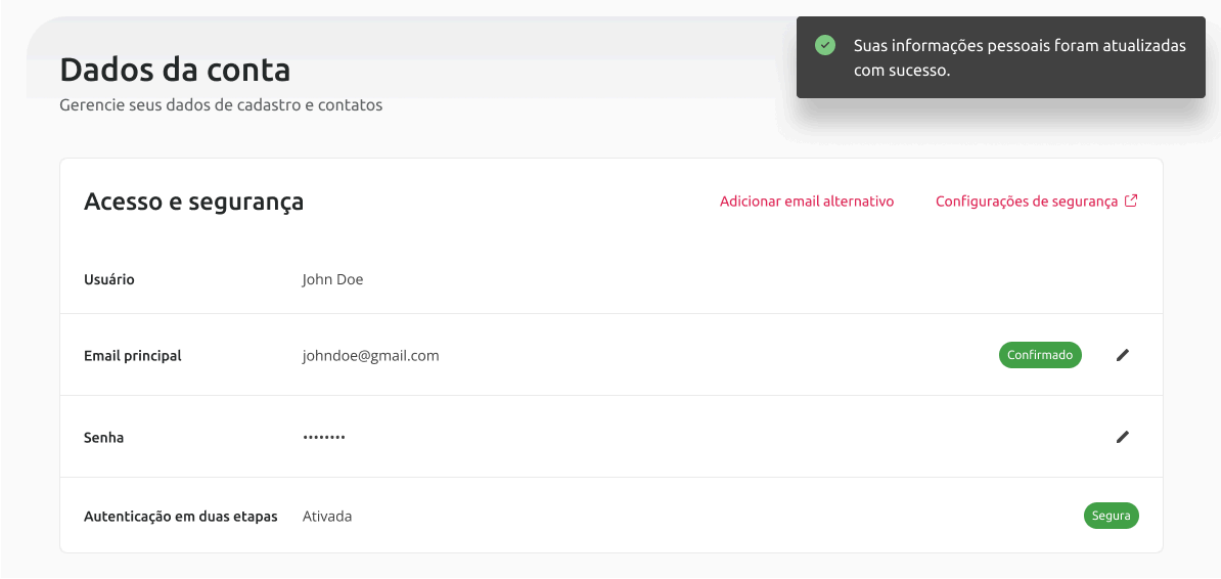


Figura 14.2: Página de Dados da conta da Central do Cliente da Locaweb. Na imagem, é mostrada a seção de "Acesso e segurança" com as informações de usuário, e-mail principal, senha e autenticação em duas etapas. Na parte superior direita, há uma notificação de fundo cinza escuro, com um ícone de "check" na cor verde e o texto na cor branca escrito "Suas informações pessoais foram atualizadas com sucesso".

Em seguida, temos a paleta de cores **warning**, formada por tons de amarelo ou laranja, cujo objetivo é chamar a atenção do usuário ou causar uma sensação de cautela.

Essa é uma paleta que pode ser confundida com a *danger* em algumas situações, e de fato elas têm propósitos parecidos, mas casos de uso diferentes. A paleta *danger* engloba os alertas de estado crítico, enquanto *warning* engloba alertas para ações em que é preciso ter cuidado, mas que não são prejudiciais.

Alguns exemplos de aplicação dessa paleta estão nos avisos de atualização de um aplicativo ou software que, para continuar sendo utilizado, exigem que a pessoa atualize obrigatoriamente; em pendências de assinatura prestes a expirar; e até mesmo em alertas

de segurança, como é o caso do Mercado Livre, que exibe uma mensagem caso existam opções de segurança não habilitadas.

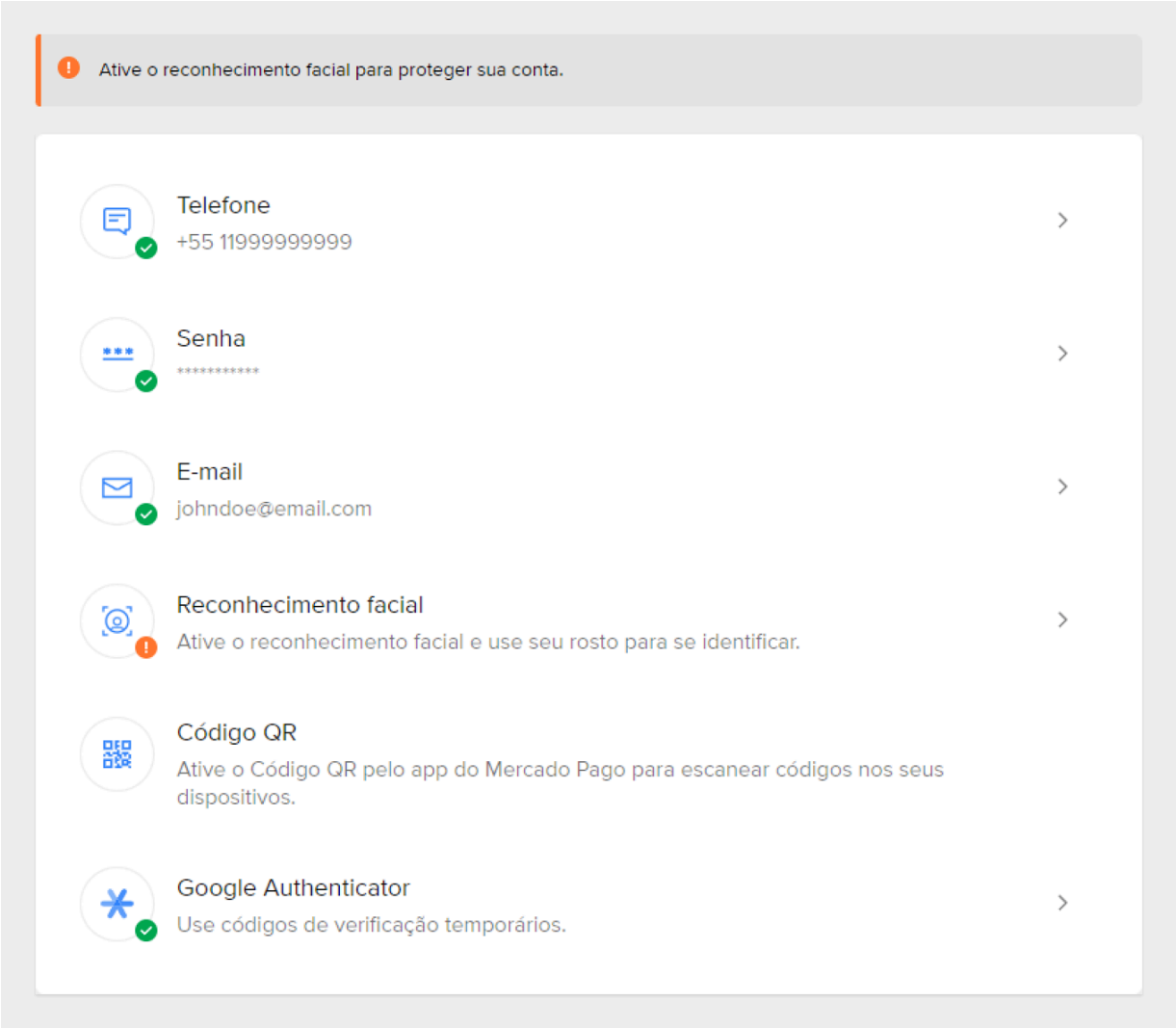


Figura 14.3: No topo, está um aviso de fundo cinza, com borda na esquerda de cor laranja. Ao lado, há um ícone redondo de cor laranja com uma exclamação no meio de cor branca, e um texto ao lado direito escrito "Ative o reconhecimento facial para proteger sua conta". Abaixo desse alerta estão os menus Telefone, Senha, E-mail, Reconhecimento facial, Código QR e Google Authenticator.

Por fim, mas não menos importante, a última paleta de cores a ser citada é a **informational**, formada por tons de azul. Seu uso é específico para mensagens ou avisos de informação geral, como esclarecimentos ou dicas.

Você poderá utilizar essa paleta principalmente em alertas ou mensagens em momentos específicos, quando houver necessidade de fornecer uma informação útil ao usuário. Por exemplo, na página "Minha Conta" da Hotmart, há um aviso no topo, antes mesmo de todos os campos do formulário, indicando que o nome público e a imagem de perfil estão visíveis para outros usuários da plataforma.

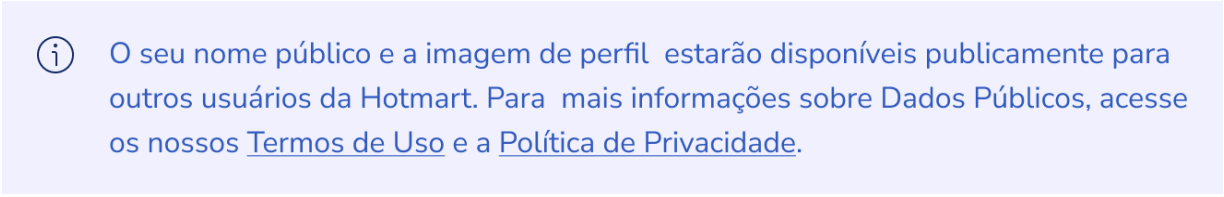


Figura 14.4: Um alerta de cor azul clara, com um ícone em formato redondo com a letra i em minúsculo no meio e um texto ao lado, ambos em azul de tom mais escuro. O texto diz "o seu nome público e a imagem de perfil estarão disponíveis publicamente para outros usuários da Hotmart. Para mais informações sobre Dados Públicos, acesse os nossos Termos e Uso e a Política de Privacidade".

Da mesma forma como comentei com as cores da marca, repito aqui: não se apegue às nomenclaturas. Esses são os termos que mais notei ser comuns, mas você encontrará design systems que usam outros nomes, como *positive* (positivo em português) e *negative* (negativo em português) nos lugares de *success* e *danger*, ou até *caution* no lugar de *warning*.

Escalas de cinza

A paleta de escalas de cinza consiste em uma variedade de tons neutros, que vão desde o branco ou cinza claro até um cinza mais escuro, e, diferente das paletas citadas anteriormente, possui uma gama de usos muito maior ao longo das páginas.

É possível utilizar diferentes escalas de cinza na composição de textos, nos planos de fundo — tanto da página quanto dos componentes — e até mesmo nas bordas.

Quando falamos especificamente de textos, é inevitável não mencionar como as escalas de cinza são benéficas para criar uma hierarquia visual adequada. Não é segredo que textos grandes são

mais evidentes do que textos pequenos, mas não apenas o tamanho serve para isso. Pensando em uma página que contém título, subtítulo e parágrafos, diferenciar esses textos entre tons de cinza que vão levemente clareando conforme a hierarquia diminui também pode auxiliar na identificação mais fácil.

Entretanto, as escalas de cinza também são muito bem aproveitadas quando falamos da composição de componentes, e em determinadas microinterações. Por exemplo, em estados desabilitados, é muito comum utilizar tons mais claros para essa representação. Em interfaces com cores mais neutras, como em campos de formulário, a composição se torna praticamente completa apenas por escalas de cinza, incluindo seus estados.

Um ponto interessante sobre a construção dessa paleta é que, por mais que o nome tenha se popularizado como "escalas de cinza", nem sempre ela segue essa designação à risca. Tornou-se comum o uso de tons de azul com baixa saturação no lugar de tons de cinza puro, pois, além de ser visualmente mais agradável, também pode reduzir a fadiga visual.

Cores literais

Todas as paletas de cores mencionadas anteriormente são importantes, pois têm casos de uso para diferentes cenários. Contudo, existem determinadas interfaces que fogem de tudo o que citamos; para esses casos, podemos utilizar as cores literais.

O uso de ilustrações coloridas e gráficos é comum em muitos produtos. Em algum momento, você precisará mais do que tons de azul, verde e escalas de cinza. Ter um grupo de paletas adicional, que contempla algumas cores a mais, como rosa, roxo e azul-esverdeado, poderá ser muito benéfico para essas situações, pois seguirá a mesma premissa das outras paletas, que é de manter a padronização.

Isso não significa que, se você já tiver uma paleta de *danger*, você deva criar outro esquema de cores em vermelho apenas para

diferenciar. Na verdade, você pode utilizar as mesmas cores, mas aplicando-as a design tokens diferentes, e até mesmo aumentando a quantidade de tons disponíveis, caso seja necessário.

Nesse sentido, a Atlassian possui uma seleção específica de cores para uso exclusivo em gráficos. A documentação sobre visualização de dados — que, por sinal, é bem interessante e recomendo a leitura — mostra como diferentes gráficos e modelos devem ser apresentados, utilizando cores tanto da marca quanto neutras, e apresenta ainda uma paleta organizada para as categorias do gráfico.

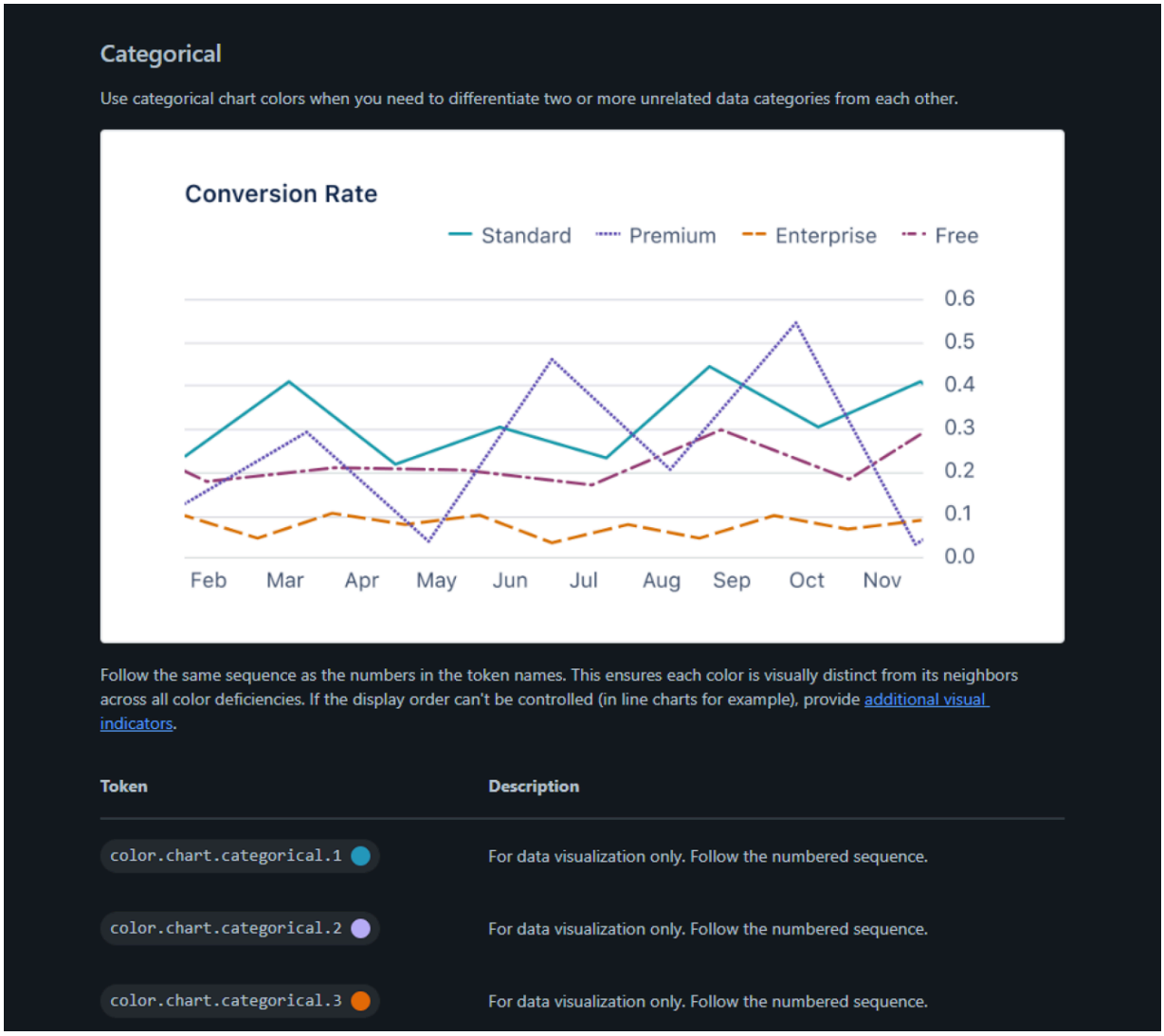


Figura 14.5: No topo da página está o título “Categórico”, e abaixo a descrição “Use cores de gráfico categóricas quando precisar diferenciar duas ou mais categorias de dados não relacionadas entre si”. Uma imagem de um gráfico de linha simulando uma taxa de conversão é apresentado, com quatro legendas, sendo a Padrão, Premium, Corporativa e Grátis. O gráfico possui linhas que começam de 0,0 e vão até 0,6, e colunas que começam em fevereiro e terminam em novembro. Logo abaixo da imagem está a seguinte descrição: “Siga a mesma sequência dos números nos nomes dos tokens. Isso garante que cada cor seja visualmente distinta de suas vizinhas em todas as deficiências de cor. Se a ordem de exibição não puder ser controlada (em gráficos de linha, por exemplo), forneça indicadores visuais adicionais”. Por último, aparece uma tabela mostrando as três primeiras linhas e duas colunas, sendo a primeira os tokens, e a segunda a descrição. As três linhas apresentam os tokens “color.chart.categorical.1”, “color.chart.categorical.2” e “color.chart.categorical.3”. As descrições são iguais: “Somente para visualização de dados. Siga a sequência numerada."

14.2 Construindo paletas

Para uma UI, é necessário ter variações de iluminação, pois é preciso oferecer uma flexibilidade e adaptabilidade ao sistema para diferentes cenários e contextos. Isso colabora diretamente com a marca, com elementos neutros, hierarquia visual e até mesmo com contrastes, garantindo a manutenção da acessibilidade.

Uma mesma paleta precisa oferecer tons que possam ser utilizados para planos de fundo, textos, ícones, bordas, microinterações e o que mais for necessário para os produtos. A questão que pode surgir nesse momento é: quantas tonalidades minhas paletas de cor devem possuir?

Uma prática que percebi ser bastante comum é que diversas equipes constroem paletas com dez ou mais tons. Acredito que isso reflita tanto o Material Design quanto outros design systems de empresas famosas. Não existe uma quantidade correta ou incorreta para construir paletas, mas alguns cuidados precisam ser tomados, principalmente quando elas forem grandes.

Como eu já comentei, toda decisão em um design system precisa ter um fundamento, e a definição de cada tonalidade não é diferente. Muitas vezes em que vi design systems com paletas de dez tons ou próximo disso, sempre havia alguns dos tons que não eram usados em nenhuma ocasião.

Por essa razão, antes da definição das suas cores, é necessário entender onde você pretende utilizá-las. Você poderá ter casos para planos de fundo, bordas, textos, ícones e vários outros elementos. Tendo isso mapeado, construir suas paletas será um trabalho mais prático, evitando futuros retrabalhos ou até mesmo informações que não serão usadas.

Para essa construção, existem inúmeras formas e ferramentas que te permitem criar paletas de cor, já pensadas para guias de estilo de

produtos digitais, desde as mais simples e rápidas até as mais complexas.

Particularmente, não acredito que exista uma maneira melhor ou pior de criar paletas, mas a forma como cada tom é construído é o que define a harmonia na interface, não apenas entre os tons, mas também em relação às outras cores.

Apesar de ter dito que não traria o foco em ferramentas neste livro, quero mencionar a seguir duas plataformas online que possuem conceitos bem diferentes, porém muito interessantes para construir paletas de cores inteiras.

Atmos

A *Atmos* (https://atmos.style/) é uma das ferramentas do *RBND studio* (https://rbnd.studio/) e oferece recursos para criar cores para interfaces digitais. Um dos recursos é voltado para design system, capaz de gerar paletas completas e permitir diversas configurações. Entretanto, o grande diferencial dela em relação a outras ferramentas é que, em vez de trabalhar com os modelos tradicionais, como HSL e HSB, ela utiliza o modelo LCH.

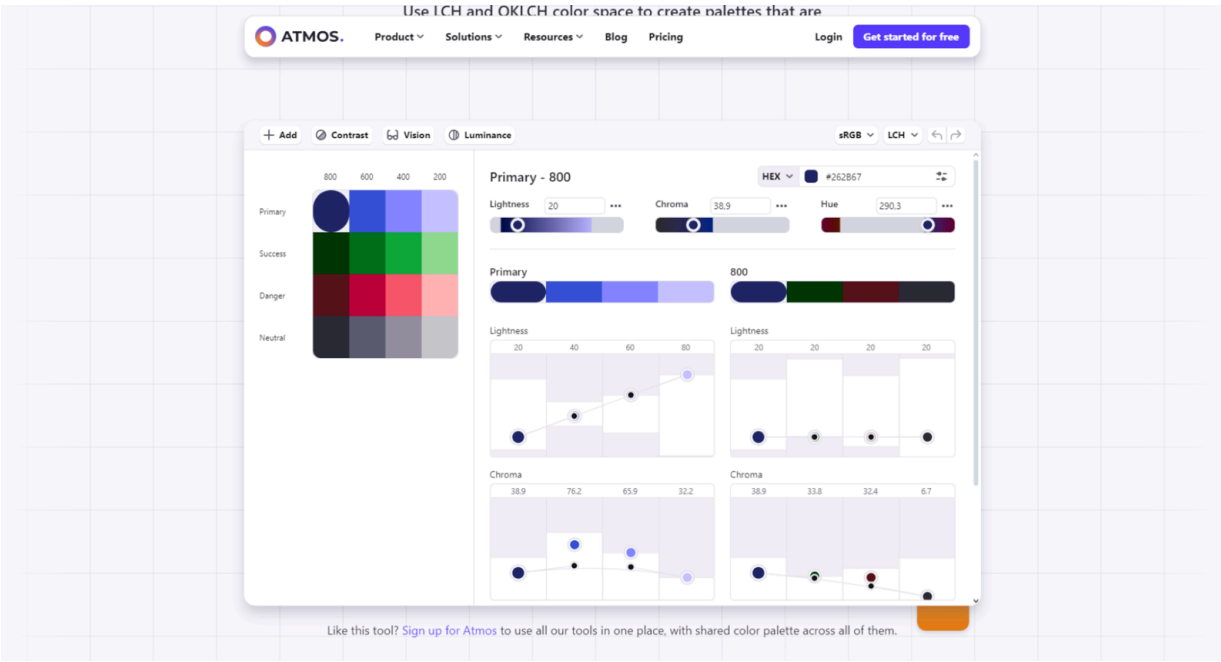


Figura 14.6: Página do Atmos que mostra a interface de construção de paletas de cor. Na esquerda, é exibida quatro paletas com quatro variações de iluminação; à direita estão as propriedades modificadores de iluminação croma e matiz.

Caso você não esteja familiarizado, o modelo LCH (*Lightness, Chroma, Hue*) é um espaço de cor derivado do modelo perceptualmente uniforme CIE LAB. Isso significa que as diferenças numéricas entre as cores correspondem de forma mais direta às diferenças percebidas pelo olho humano. Isso ajuda a garantir que as cores escolhidas sejam visualmente equilibradas e harmoniosas.

Esse modelo representa as cores de acordo com três componentes:

- *Lightness* (Claridade), que vai de 0 (preto) a 100 (branco);
- *Chroma* (Croma), que indica a intensidade ou vivacidade da cor;
- *Hue* (Matiz), que especifica a tonalidade em graus no círculo cromático (0-360).

Uma das vantagens de se trabalhar com esse modelo, que a própria ferramenta não só facilita como também evidencia, está na otimização dos contrastes. Diferente de outros modelos, que podem gerar grandes diferenças de contraste nos mesmos níveis de

iluminação, o LCH mantém as proporções idênticas ou extremamente próximas.

Inclusive, caso você tenha definido uma paleta, mas percebeu que havia a necessidade de aumentar ou até mesmo diminuir as variações de iluminação, isso não se torna um problema. Os contrastes não serão perdidos e, a partir das configurações que você definir, a ferramenta garante que as proporções dos tons serão mantidas.

ColorBox

O *ColorBox* (https://colorbox.io/), criado por Kevyn Arnott, é uma ferramenta que já passou por algumas evoluções e também tem o foco em interfaces digitais, colaborando significativamente na construção de paletas para design systems.

O modelo de cor que a ferramenta usa é o HSB (*Hue, Saturation, Brightness*), muito mais conhecido do que o LCH, da ferramenta citada anteriormente. Contudo, o grande destaque da ColorBox é o conceito de construção das variações de iluminação baseado nas curvas de Bézier.

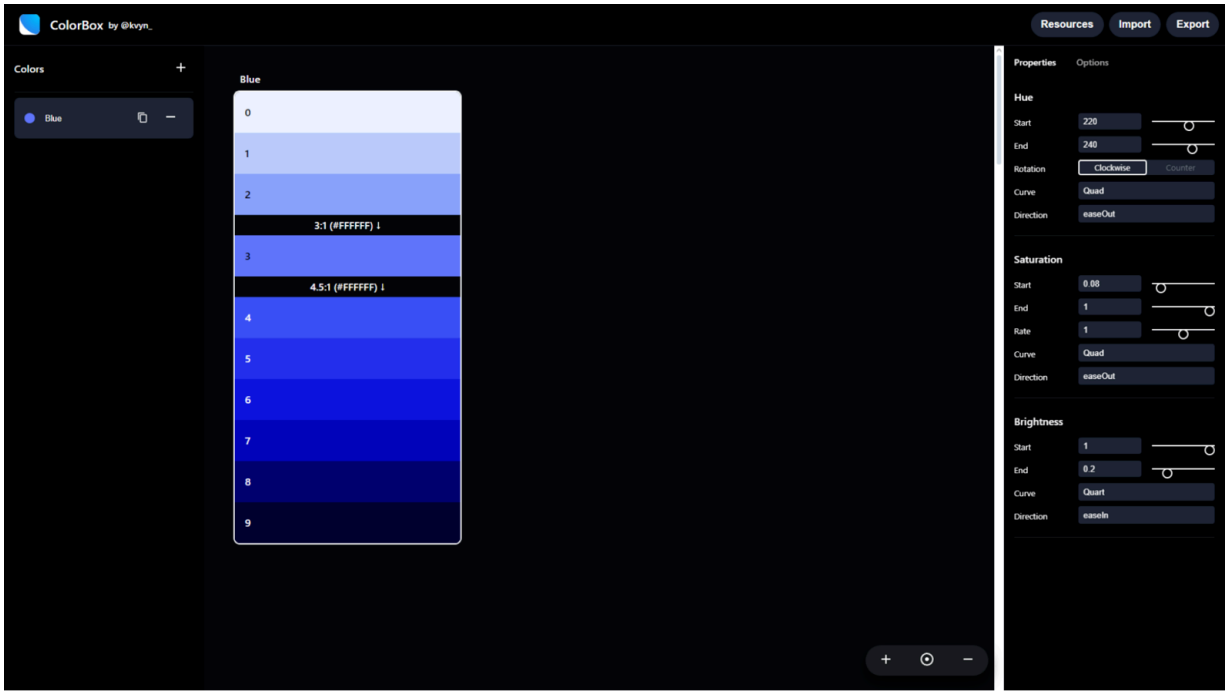


Figura 14.7: Página do ColorBox que mostra a interface de construção de paletas. À esquerda, é exibido a paleta de cor azul cadastrada. Ao meio, está a exibição da paleta, com dez variações de iluminação. Já à direita, estão as propriedades modificadoras de matiz, saturação e brilho.

Muito conhecidas por editores de vídeo e motion designers, as curvas de Bézier são ferramentas matemáticas utilizadas em gráficos por computador, design e animação para modelar formas suaves e precisas. Elas são definidas por pontos de controle, que determinam tanto a direção quanto a forma da curva.

Eu sei, na primeira vez que você se depara com isso, pode ser um pouco confuso, já que não é o tema mais simples de todos, nem está tão relacionado a cores quanto a outros elementos, como animações. É justamente por isso que considero incrível a teoria desenvolvida para gerar as paletas.

A ferramenta permite que você identifique os valores de início e de fim, tanto da matiz quanto da saturação e brilho. E, a partir dessas definições, ambas abrem a possibilidade para que você controle a curva e a direção que ocorrerá em cada uma delas.

A imagem seguinte, tirada do próprio artigo da ferramenta, exemplifica melhor como isso acontece:

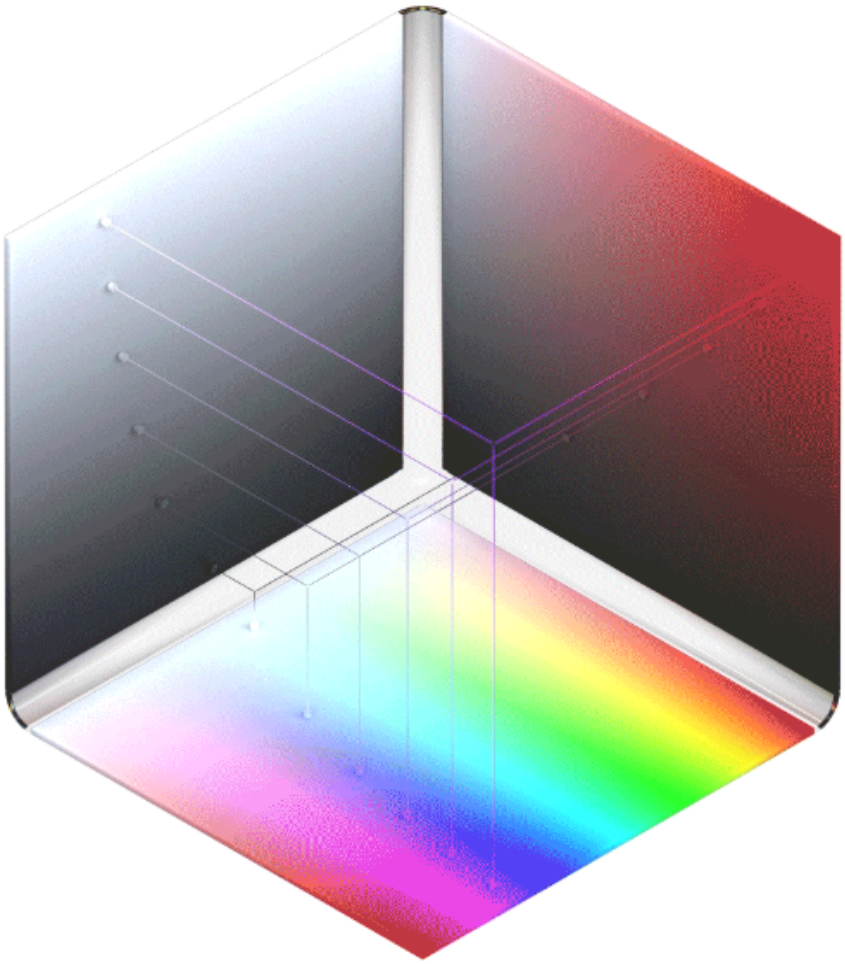


Figura 14.8: Uma representação 3D de um cubo mostra na face da esquerda o brilho e na direita a saturação. Na parte inferior, há a matiz. Seis traços saem de cada uma das faces e se encontram no centro do cubo, demonstrando qual variação de cada uma das faces está sendo utilizada para criar a paleta de cor.

As definições básicas de "HSB" determinam os valores de matiz, saturação e brilho, indicando onde cada um começa e termina, enquanto as curvas são responsáveis por cruzar essas informações. A partir da quantidade de variações que você definir, as paletas serão geradas de acordo com essas definições.

Mesmo que a ferramenta permita que você se baseie em alguma cor específica, liberando um campo de hexadecimal, ela o torna independente da decisão de cores, pois as definições de valores e curvas geram a paleta automaticamente. Isso também é vantajoso para a criação do modo escuro, pois, com as definições já estabelecidas, a própria ferramenta cria a nova paleta sem necessidade de novas configurações.

CAPÍTULO 15
Elevação

Uma das formas de pensar a construção de interfaces digitais é imaginá-las como planos tridimensionais, criando uma ilusão de profundidade no layout, em vez de simplesmente serem interfaces 2D. Nesse modelo, devemos observar os elementos de uma página como se estivessem sobrepostos, como vários papéis recortados que se encaixam nesse espaço, formando uma página completa.

Assim como no mundo real, ao sobrepor diferentes recortes de papel, tiramos um pouco o foco dos que estão atrás, destacando apenas os que estão mais próximos de nós. Ao posicionar esses papéis sob um ponto de iluminação, aproximando-os ou afastando-os uns dos outros, percebemos diferentes intensidades de sombra em cada camada.

Cada um desses detalhes está presente em nosso cotidiano, tanto no mundo real quanto no digital, no que chamamos de **elevação**.

15.1 Elevação e hierarquia

A elevação é a aplicação de efeitos de sombra em interfaces digitais. Não deve ser entendida somente como um recurso visual, mas também como um atributo de hierarquia e interação. Na prática, componentes que utilizam sombras parecem flutuar sobre os demais elementos, tendo o plano de fundo da página como a superfície mais baixa.

Essa visão tridimensional é normalmente pensada somente pelas pessoas que estão construindo as páginas. Para o usuário que está navegando, o benefício real da elevação está na hierarquia.

Imagine cada componente de um design system como uma camada nesse plano tridimensional. Alguns deles serão colocados lado a lado sobre a superfície, enquanto outros serão dispostos acima de outros componentes. Como eu comentei antes, quanto mais próximo de nós o elemento estiver, mais foco ele terá, e quanto mais afastado estiver da base, maior será a intensidade da sombra do componente.

Com esse recurso, níveis diferentes de sombras são criados para transmitir hierarquia e reforçar a organização dos componentes na página. O Google trouxe esse conceito muito bem aplicado na segunda versão do Material Design, criando até mesmo um diagrama do funcionamento do seu sistema de elevação em seus produtos:

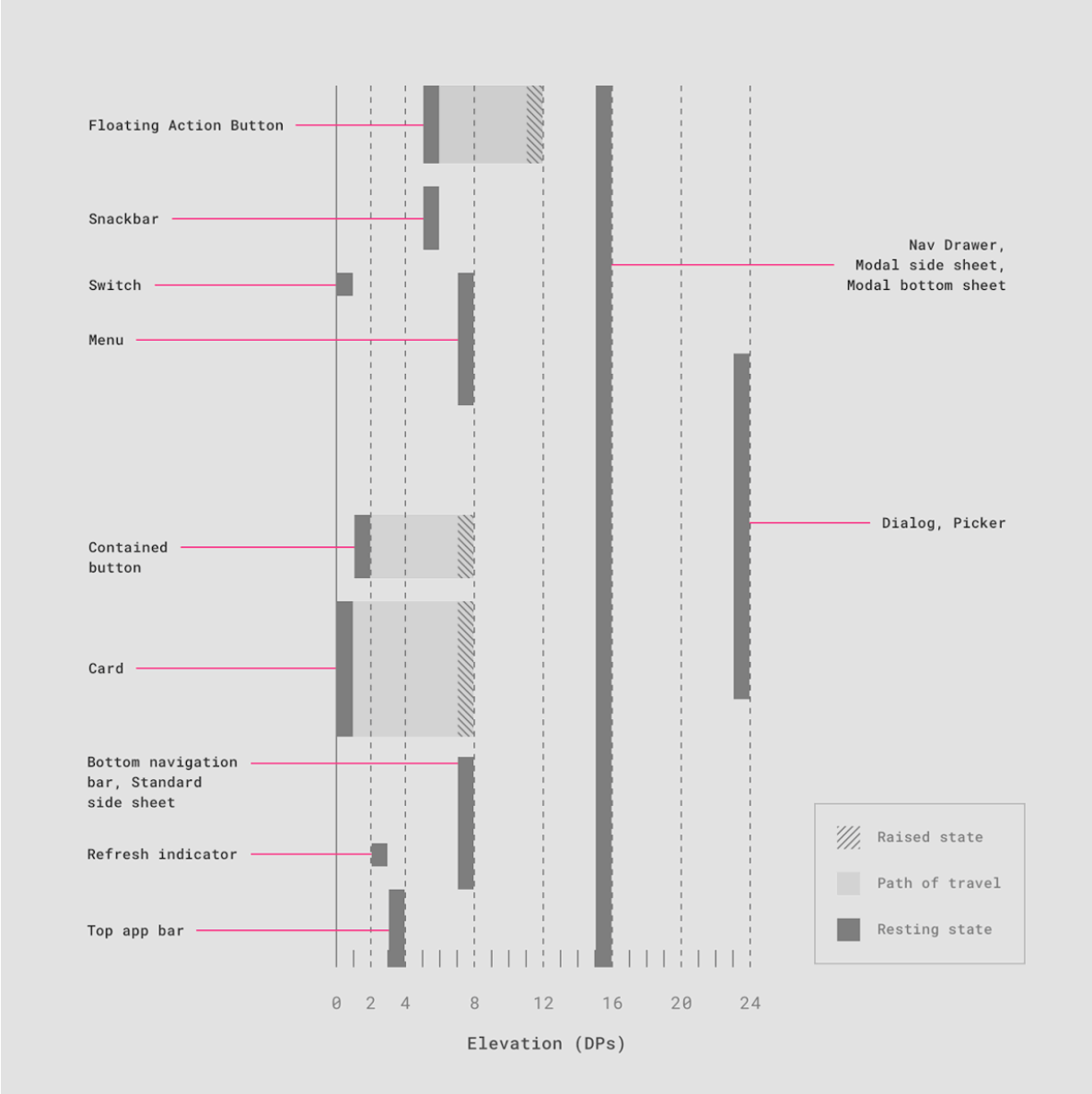


Figura 15.1: Diagrama ilustrando como a elevação de elementos afeta o comprimento e a suavidade da sombra em uma largura de 600 DP. Linhas pontilhadas mostram diferentes ângulos de projeção de sombra para diferentes níveis de elevação.

No Android, principal produto do Google que usa o Material Design, adotou-se a ideia de uma luz posicionada no topo da tela dos smartphones, a `600dp` (*density-independent pixels*) de distância. A posição dos componentes define o deslocamento de suas sombras: elementos próximos à borda inferior do dispositivo apresentam deslocamento no eixo Y, ou seja, vertical.

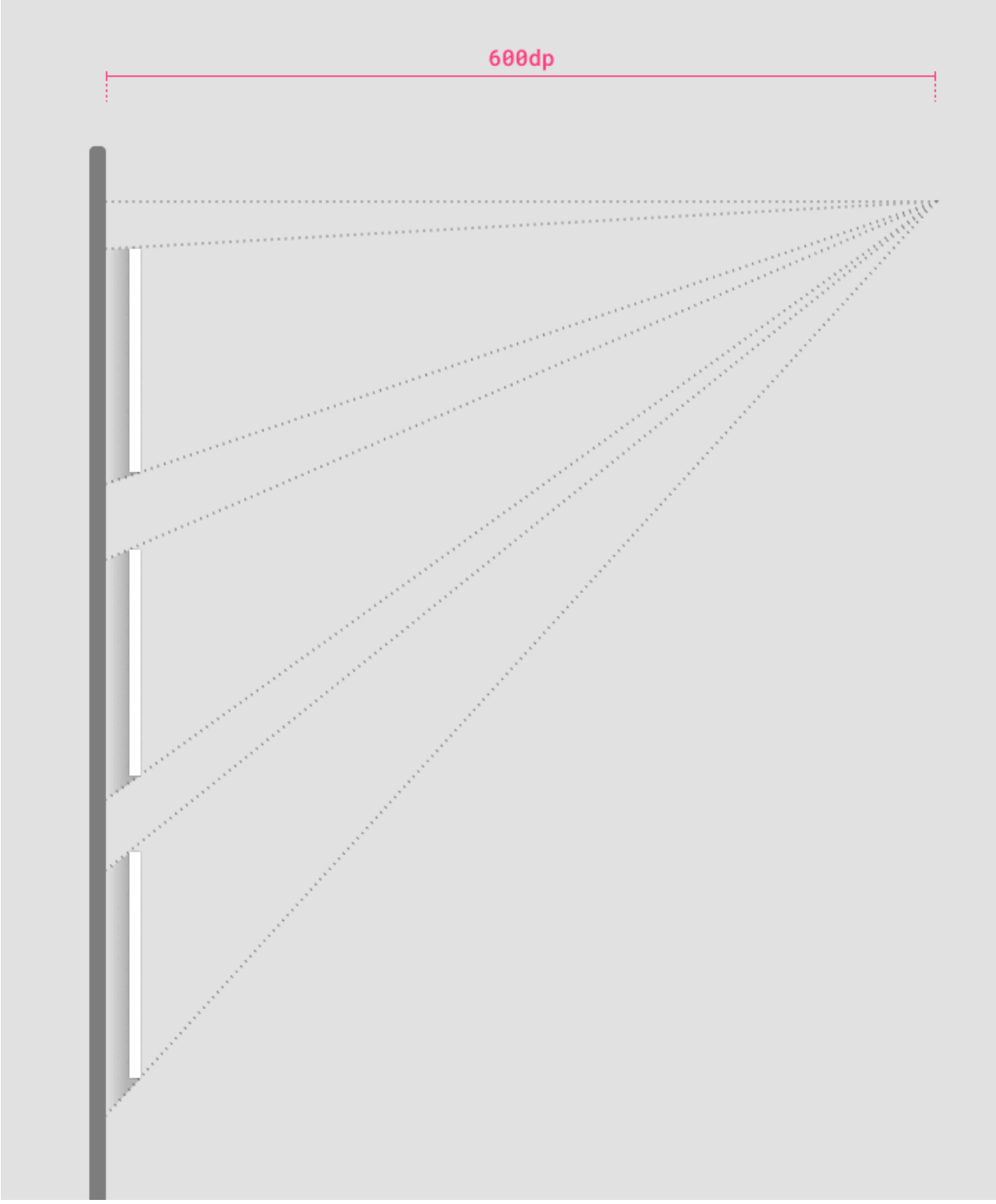


Figura 15.2: Diagrama mostrando um mastro cinza vertical à esquerda com linhas diagonais pontilhadas saindo em direção ao canto superior direito, formando triângulos que representam diferentes ângulos de abertura. No topo, há uma medida horizontal em rosa indicando 600 dp de largura.

15.2 Elevação dinâmica

Embora o principal objetivo das sombras seja dar foco e definir a hierarquia entre os componentes da página, elas não foram pensadas apenas para serem estáticas.

Temos uma série de componentes que exigem alguma interação do usuário. Botões e cards podem executar alguma ação após o clique ou toque. Elementos como esses podem ser definidos com elevação dinâmica e terão uma mudança de eixo (normalmente no eixo Y), subindo no plano.

Em elementos interativos, a sombra é um recurso valioso para indicar ao usuário que determinado item pode ser clicado ou manipulado. Esse recurso vem sendo utilizado há décadas por empresas de tecnologia, como a Microsoft nas primeiras versões do Windows, em janelas, botões, menu Iniciar e outros elementos clicáveis.

Figura 15.3: Tela inicial do Windows 98 com interface clássica em verde água. Uma janela chamada "Welcome to Windows 98" apresenta um menu com tópicos como Register Now, Connect to the Internet, Discover Windows 98 e Maintain Your Computer. Há botões, ícones de desktop como Meu Computador e Meus Documentos, além da barra de tarefas na parte inferior.

15.3 Aplicando elevações

O primeiro passo é identificar quais componentes exigem elevação, ou seja, aqueles que, no plano tridimensional da página, não estão colados à superfície. Se fizer sentido no seu design system, considere também os componentes que podem acabar tendo mudanças de elevação durante o uso. Para facilitar, segue uma lista de alguns componentes que podem fazer parte disso:

- Alert
- Button
- Card
- Drawer
- Dropdown
- Menu
- Modal
- Notification
- Popover
- Tooltip

O passo seguinte é analisar a função de cada componente e como ele se apresenta na página, para então definir sua ordem de importância. Os modais, por exemplo, ocupam toda a tela e bloqueiam a navegação até serem concluídas. Por isso, e conforme indicam referências de outros design systems, elas possuem maior relevância e, por estarem sobre toda a interface, projetam sombras mais intensas.

Como passamos a imaginar um produto digital em três dimensões, em vez de duas, é interessante definirmos valores para atuar com camadas. Considere a divisão que foi definida com os componentes para classificar a quantidade de camadas. Apenas lembrando: a camada "0" é a base ou superfície que não leva sombra.

Por fim, é necessário decidir as propriedades visuais da sombra. Para isso, a gente precisa considerar cinco propriedades:

1. `Offset X` : deslocamento horizontal;
2. `Offset Y` : deslocamento vertical;
3. `Blur Radius` : controla o desfoque da sombra;
4. `Spread Radius` (opcional): expande ou contrai o tamanho da sombra;
5. `Cor` : define a cor e a opacidade da sombra.

O deslocamento no eixo Y é o principal atributo para causar o efeito de elevação, pois combina o desfoque e a opacidade da sombra,

dando a impressão de estar levitando sobre a superfície. Deslocamentos no eixo X são menos usados, mas podem ser observados em componentes que se movimentam na horizontal ou ficam fixos em algum canto da tela.

Por mais que um único estilo de sombra exija pelo menos quatro dos cinco atributos, é possível expandir para várias sombras dentro de um único estilo. Na prática, trabalhando com CSS, cada sombra é separada por vírgula da seguinte forma:

```
box-shadow:
  0px 1px 2px rgba(0, 0, 0, 0.12),
  0px 4px 8px rgba(0, 0, 0, 0.08);
```

Nesse comando, a primeira sombra é menor, mais próxima do objeto, usada mais para definir um contorno, enquanto a segunda é maior, mais difusa, criando a sensação de flutuação. Essa aplicação pode ser bem eficaz para dar mais realidade a um efeito de sombra, pois objetos na vida real podem ser atingidos tanto por iluminação natural quanto por alguma iluminação direcional.

Com esses conceitos, é possível definir diferentes escalas para todos os componentes, independentemente de ter ou não interação, ou do seu foco em uma página.

CAPÍTULO 16
Espaçamento

Quando falamos dos fundamentos das interfaces digitais, é impossível não falar também sobre o espaçamento, pois mesmo sendo um dos elementos mais silenciosos de uma página, ele continua sendo um dos pilares.

O espaçamento se destaca por sua capacidade de estruturar a interface, dar ritmo ao layout e influenciar diretamente a clareza e a experiência do usuário. É por meio dele que conseguimos criar respiros, definir relações entre elementos e guiar o olhar de forma natural.

Mesmo sem chamar atenção direta, uma interface com espaçamento bem aplicado se torna mais funcional, legível e harmoniosa. Para iniciarmos esse assunto, quero ressaltar três tópicos que tornam esse atributo tão valioso para produtos digitais.

Primeiramente, temos a **legibilidade**, que quando bem aplicada, resulta em um ótimo "respiro" entre textos, botões e componentes. Isso torna a leitura mais fluida, evita que o conteúdo pareça poluído e melhora a experiência de navegação. Além disso, espaços adequados em áreas clicáveis aumentam a acessibilidade e reduzem a chance de erros de toque, especialmente em dispositivos móveis.

Além disso, o espaçamento é uma ferramenta essencial para criar uma **hierarquia visual** na interface. Ele ajuda a mostrar claramente quais elementos estão relacionados entre si e quais pertencem a seções diferentes. Espaçamentos maiores separam blocos distintos, enquanto os menores mantêm grupos de elementos unidos. Isso guia o olhar do usuário e facilita a compreensão da estrutura da página, mesmo sem o uso de cores ou linhas divisórias.

Por fim, temos a **escalabilidade**, uma das que considero mais importantes do design de interface. Padronizar o espaçamento por meio de uma escala definida ajuda a manter a interface coesa e profissional. Isso evita variações desnecessárias entre telas e componentes, facilita o trabalho em equipe e acelera o processo de design e desenvolvimento.

16.1 Propriedades de espaçamento

Ao analisar uma página, percebemos uma série de respiros entre os elementos, seja nas laterais que centralizam o conteúdo, entre um rótulo e um componente ou até mesmo dentro do próprio componente, como um campo de texto, que possui um pequeno espaço entre o texto digitado e as bordas ao redor. Cada um desses que eu citei colabora para manter a harmonia, mas, na prática, são aplicações diferentes: o *margin*, o *padding* e o *gap*.

Margin

Usamos o *margin* quando queremos afastar um elemento de outro, como no caso de um título que possui 4 pixels de distância do texto a seguir:

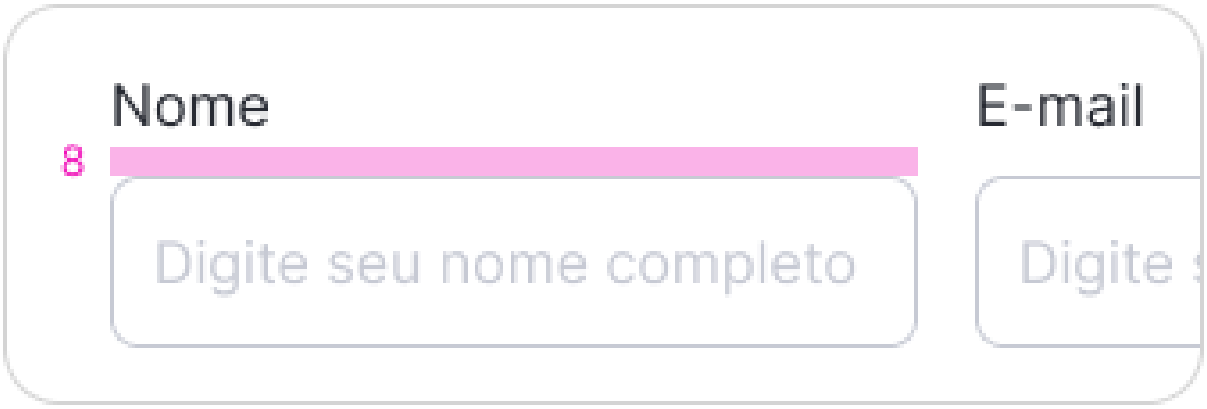


Figura 16.1: Um campo de texto com o rótulo "Nome" e o placeholder "Digite seu nome completo". Entre o rótulo e o campo está uma marcação rosa com o número "8", indicando uma margem de 8 pixels.

Padding

O *padding* é usado para criar um respiro interno, ou seja, para definir o espaço dentro do próprio elemento. Um dos maiores exemplos disso são os botões, quando definimos o espaço entre o texto do botão e suas bordas.

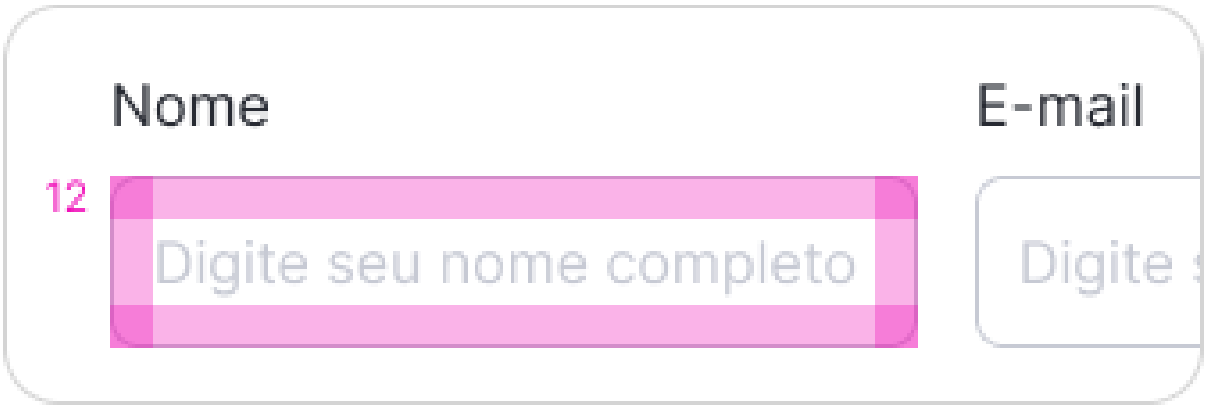


Figura 16.2: Um campo de texto com o rótulo "Nome" e o placeholder "Digite seu nome completo". Dentro do campo há uma marcação rosa em cada lado e o número 12 ao lado, indicando um padding de 12 pixels.

Gap

O *gap*, semelhante ao *margin*, também define a distância entre um elemento e outro; contudo, ele só é aplicado em layouts flexíveis,

aplicando espaçamento entre colunas e linhas. É utilizado em layouts com *flexbox* aplicado.

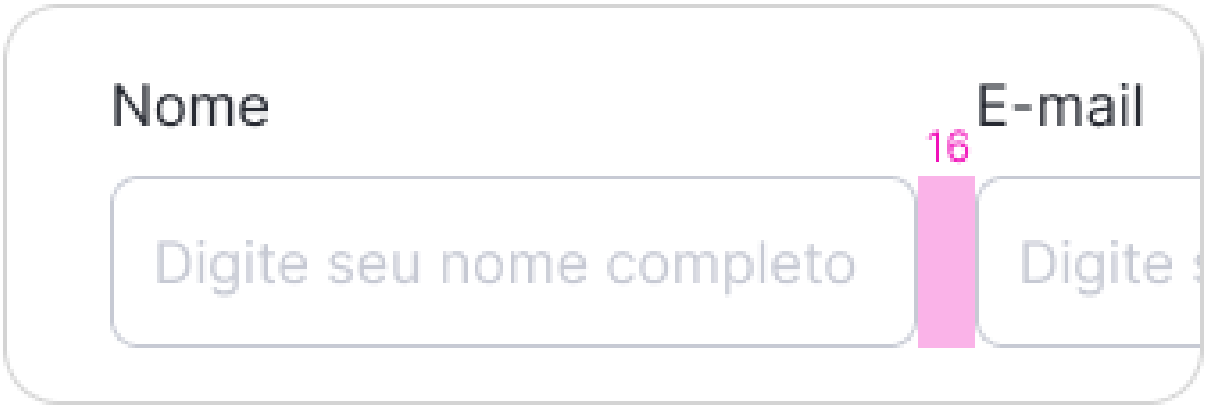


Figura 16.3: Dois campos, sendo um de "Nome" e outro de "E-mail", ambos na horizontal. Entre eles, há uma marcação rosa com o número "16", indicando um gap de 16 pixels.

16.2 Unidades de medida

Quando falamos de espaçamento, estamos tratando da distância entre os elementos da interface. Para isso, consideramos uma série de valores, desde os menores, para respiros mais curtos, até os maiores. De forma totalmente natural, é comum pensarmos nessa distância em pixels, já que é a unidade de medida básica em qualquer ambiente digital, pois toda tela é composta por pixels.

Contudo, o pixel não é a única unidade de medida existente para trabalharmos. É importante que você não só conheça as outras medidas disponibilizadas no *CSS*, como também saiba trabalhar corretamente com elas, principalmente se você não tem vivência com código.

Pixel

Pixel é a unidade fixa mais usada em *CSS*, que representa um ponto na tela do dispositivo. É uma medida absoluta, ou seja, sempre vai ter o mesmo tamanho independente do elemento ou do tamanho da

fonte. Por isso, é muito usada quando você quer controle exato sobre o espaçamento, tamanho ou posicionamento.

REM

REM (*root em*) é uma unidade que representa o tamanho da fonte raiz do documento, normalmente definido no elemento `<html>`. Por padrão, esse tamanho é 16 pixels na maioria dos navegadores, mas pode ser alterado tanto nas configurações quanto pelo próprio programador durante o desenvolvimento do site. Dessa forma, quando é utilizado `1rem` para definir um espaço, o navegador automaticamente traduz para 16 pixels.

Essa unidade é muito útil para garantir consistência e acessibilidade. Seu uso é tanto para espaçamentos no geral quanto para tipografia, principalmente em tamanhos de texto. Falaremos mais sobre essa medida no capítulo 17.

EM

EM é a unidade que se baseia no tamanho da fonte do próprio elemento ou do seu elemento pai. Isso significa que o valor muda dependendo de onde está aplicado. Por exemplo, se a fonte raiz de um componente, como um *card*, for definida como 14 pixels, sempre que for aplicado um tamanho de `1em` neste card, automaticamente o navegador entenderá como 14 pixels. Outros elementos identificados com tamanhos diferentes na raiz não irão impactar os valores `em` definidos nesse *card*.

Porcentagem

A porcentagem é uma unidade que calcula o valor com base no tamanho do elemento pai. Ela permite criar layouts e espaçamentos proporcionais, se adaptando automaticamente conforme o tamanho do pai muda. Nesse caso, se considerarmos uma interface com 400

pixels de largura, e aplicarmos 50% em um dos elementos dentro dessa interface, ele passará a ter 200 pixels de largura.

Viewport Height e *Width*

As unidades `vw` e `vh` são usadas para criar layouts que se adaptam ao tamanho da tela do usuário, com base no tamanho da *viewport*, que é a área visível do navegador. São medidas ideais pra criar elementos que se ajustam ao tamanho da tela, como seções em tela cheia, textos responsivos ou espaçamentos que acompanham a *viewport*.

Considerando que `100vh`, por exemplo, representa a altura total do *viewport*, quando indicamos que um elemento terá `50vh`, automaticamente ele sempre terá metade da altura da tela, independentemente do dispositivo que o usuário estiver acessando.

16.3 8-point grid

Aplicar espaçamentos em um produto digital vai muito além do olhômetro, ou seja, de testar valores aleatórios até sentir que ficaram bons. O sistema de espaçamento de um design system também precisa ser escalável; da mesma forma que as cores, elevações ou escalas tipográficas.

Um dos métodos mais utilizados para aplicar espaçamentos é trabalhar com múltiplos de 4 (4, 8, 12, 16…) ou de 8 (8, 16, 24, 32…) pixels, pois são perfeitamente divisíveis e fáceis de serem refinados, até mesmo para valores menores (1 ou 2px), caso seja necessário para cenários específicos.

A razão pela qual se utilizam múltiplos de 4 ou 8 também está relacionada à largura das telas de monitores ou dispositivos móveis que, caso você não tenha reparado, são, na maioria das vezes, divisíveis por esses valores. Quando uma página é projetada com

essa simples matemática em mente, todos os elementos garantem consistência visual, ficando devidamente organizados.

Na tabela a seguir, você pode conferir oito das resoluções mais comuns (em computadores) e quais delas são e não são divisíveis por 8.

Resolução	**Múltiplo de 8**
1366x768	Não
1920x1080	Sim
1280x1024	Sim
1440x900	Sim
1600x900	Sim
1280x800	Sim
1024x768	Sim
1360x768	Sim

Mas não se engane: por mais que a regra seja aplicar valores múltiplos de 4 ou 8 em *margin*, *padding* e outros atributos de espaçamento, é importante que você não force determinados objetos. Já revisei alguns projetos em que a altura do botão estava com 48 pixels, mas com 10 pixels de *padding* nos lados superior e inferior, pois a pessoa acreditava que a altura não poderia fugir dessa regra.

Botões, assim como a maioria dos componentes, não possuem valores estabelecidos para largura ou altura, sendo variáveis a partir dos atributos de espaçamento aplicados. Normalmente, vamos fixar um valor múltiplo de 4 ou 8 a uma largura ou altura quando

queremos manter a consistência no tamanho de elementos que repetem algumas vezes na página.

Caso você queira se aprofundar mais nesse assunto, você pode buscar pelos termos “4-Point Grid” ou “8-Point Grid”. Como recomendação, sugiro a leitura do artigo *The 8-Point Grid* (https://spec.fm/specifics/8-pt-grid).

Capítulo 17
Tipografia

A tipografia é um dos pilares fundamentais de qualquer design system. Muito mais do que apenas definir uma fonte como padrão, ela determina como a informação será lida, percebida e compreendida pelos usuários. Uma boa estrutura tipográfica garante consistência visual, hierarquia clara e melhora a legibilidade em diferentes interfaces, dispositivos e contextos.

Uma tipografia bem definida ajuda a criar identidade, orientar o olhar e transmitir a personalidade da marca. Ela estabelece ritmo, proporção e equilíbrio visual, guiando a pessoa usuária pela interface de forma natural. Quando usada de forma consistente, a tipografia se torna uma ferramenta poderosa para reforçar confiança, clareza e acessibilidade em toda a experiência digital.

Pequenos detalhes, quase invisíveis, fazem toda a diferença na leitura, no ritmo e na experiência geral. Entender como esses elementos se conectam ajuda a construir interfaces mais harmoniosas e funcionais, onde cada palavra ocupa o espaço certo e comunica com clareza.

17.1 Definição da fonte

Antes de nos aprofundarmos nesse assunto, é importante entender uma coisa: tipografia e fonte são conceitos diferentes, mas que estão relacionados.

Uma fonte nada mais é do que um arquivo digital com uma coleção de caracteres, como letras, números e símbolos que seguem o mesmo estilo visual. Já a tipografia é o conjunto de escolhas e regras que definem como essas fontes serão usadas em um projeto, considerando aspectos como tamanho, peso, espaçamento, alinhamento e hierarquia, para garantir uma leitura eficiente e uma comunicação visual coerente.

A primeira decisão que precisa ser tomada ao pensar em tipografia é sobre qual fonte será utilizada para compor as páginas dos produtos. Essa decisão pode ser muito simples se considerar o que você leu sobre consistência da marca, no capítulo 8. Muitas empresas possuem não apenas uma identidade visual consistente, mas também um manual de marca detalhado, que inclui, entre outros aspectos, as especificações relacionadas à tipografia.

Entretanto, nem todas as empresas possuem algo assim. Esse ainda pode ser um problema simples de resolver, desde que o logotipo utilize uma fonte adaptada para a web. Porém, é importante considerar que existem marcas cujas letras ou caracteres do logotipo não derivam de uma fonte, sendo projetados especificamente para aquela identidade — como no caso do logotipo da Alura. Nesses casos, cabe aos responsáveis pela construção do design system pesquisar e validar as possibilidades.

A característica mais importante na escolha de uma fonte é na sua leiturabilidade, ou seja, a facilidade que uma pessoa tem de entender alguma informação textual conforme ela lê. No entanto,

esse não é o único fator decisivo. É imprescindível que a fonte contenha não apenas letras, mas também números, pontuações e símbolos. Ao considerar a construção de um design system brasileiro, é necessário contemplar os acentos, pois erros de português não apenas comprometem a imagem do produto e da marca, como também dificultam a compreensão para pessoas que utilizam softwares de leitura de tela.

Outro fator que deve ser decisivo durante a escolha é a variedade de pesos. Hoje, é comum encontrarmos fontes que possuem variações desde a mais fina, chamada "thin", até a mais espessa, chamada "black", um movimento natural de evolução da tipografia.

No contexto do design system, dificilmente você usará ou terá casos de uso para todos os pesos, principalmente se considerarmos os casos que podem prejudicar a leiturabilidade, como o peso "thin", usado em textos pequenos. O mínimo esperado é que a fonte ofereça, pelo menos, os pesos "regular" e "bold". Caso seja necessário ampliar essa variedade, é importante validar os pesos para garantir que cada um seja facilmente distinguível dos demais.

17.2 Hierarquias e escalas

Todo produto é composto por hierarquias diferentes de textos, que não só servem para comunicar algo, mas para guiar o usuário pela página conforme a sua leitura. Aprendemos essas hierarquias desde crianças com os livros, e não é difícil entender e diferenciar que títulos são os maiores textos de uma página; os subtítulos, um pouco menores; e os parágrafos, com tamanhos mais reduzidos ainda.

Quando tratamos especificamente de produtos digitais, lidamos com páginas longas que reúnem diferentes tipos de informação e que

precisam de destaques individuais, sem perder a hierarquia adequada de importância.

Não há algo que indique quais modelos de texto devem existir e, claro, que não existem obrigatoriedades dentro de um design system. Contudo, é extremamente comum vermos duas categorias: **body** e **heading**.

Body texts, ou "textos de corpo" (em português), nada mais são do que os textos usados para o conteúdo principal de uma página.

O melhor exemplo que posso citar é o parágrafo — exatamente como este que você está lendo agora. Já os *headings*, ou cabeçalhos, são os textos usados em títulos, subtítulos e outras informações que chamam a atenção durante a leitura, sendo, normalmente, a primeira coisa que o usuário observa.

Se você realizou o inventário de interfaces de que falamos no capítulo 2, então é provável que você já tenha uma base de quais são as categorias e as escalas textuais do seu produto. Não se preocupe se forem várias, pois não existe um mínimo nem um máximo. O Atlassian design system, por exemplo, possui nove escalas diferentes, já o Material Design, apenas cinco, mas cada uma delas possui três variações de tamanho.

Uma das formas com que já trabalhei para manter harmonia e consistência entre as escalas — e que também pode ser encontrada na documentação de alguns design systems, como o da Atlassian — é o uso do conceito de **escalas modulares**.

Escalas modulares

As escalas modulares — ou *modular scale*, se você preferir em inglês — são uma abordagem fundamental e muito conhecida na tipografia. Consistem em um sistema de proporções para criar harmonia visual para diferentes escalas de texto. Em vez de escolher tamanhos aleatórios, você usa uma razão matemática para definir os tamanhos.

Esse conceito é bastante conhecido entre profissionais de UI e tornou-se comum em design systems, embora não seja um estudo recente. Há mais de 30 anos, o tipógrafo Robert Bringhurst já discutia esse tema em seu livro *"Os Elementos do Estilo Tipográfico"*. A ideia apresentada pelo autor é que tamanhos de fonte, entrelinhas, margens e espaçamentos devem manter uma relação harmônica entre si, estabelecendo uma conexão entre tipografia e música. Assim, boas composições tipográficas criam um ritmo visual semelhante ao de uma escala musical bem construída.

Para criar as escalas, um simples cálculo de multiplicação e divisão é feito para definir as hierarquias superiores e inferiores. Mas, antes que isso seja feito, é necessário tomar uma decisão inicial referente ao tamanho da fonte raiz. Páginas web naturalmente utilizam o valor 16 pixels como padrão, sendo esse também um valor extremamente recomendável para ter uma leitura confortável, e esse pode ser o seu ponto de partida.

A partir disso, é necessário somente decidir qual proporção (também mencionada como razão) utilizar para criar todas as escalas tipográficas. As proporções são razões matemáticas específicas, sendo que muitas delas vêm da música, como Terça Menor ou Sexta Maior, que servem para gerar a sequência de tamanhos dos textos. É por meio de simples multiplicações e divisões dessa proporção em relação à altura do texto que se gera a escala padronizada.

É claro que você não precisa fazer esses cálculos manualmente, e nem mesmo decorar quais são as proporções existentes. Há uma série de sites que geram essas escalas modulares, e como recomendação eu deixo o site *Modular Scale* (https://www.modularscale.com/?1&em&1.5).

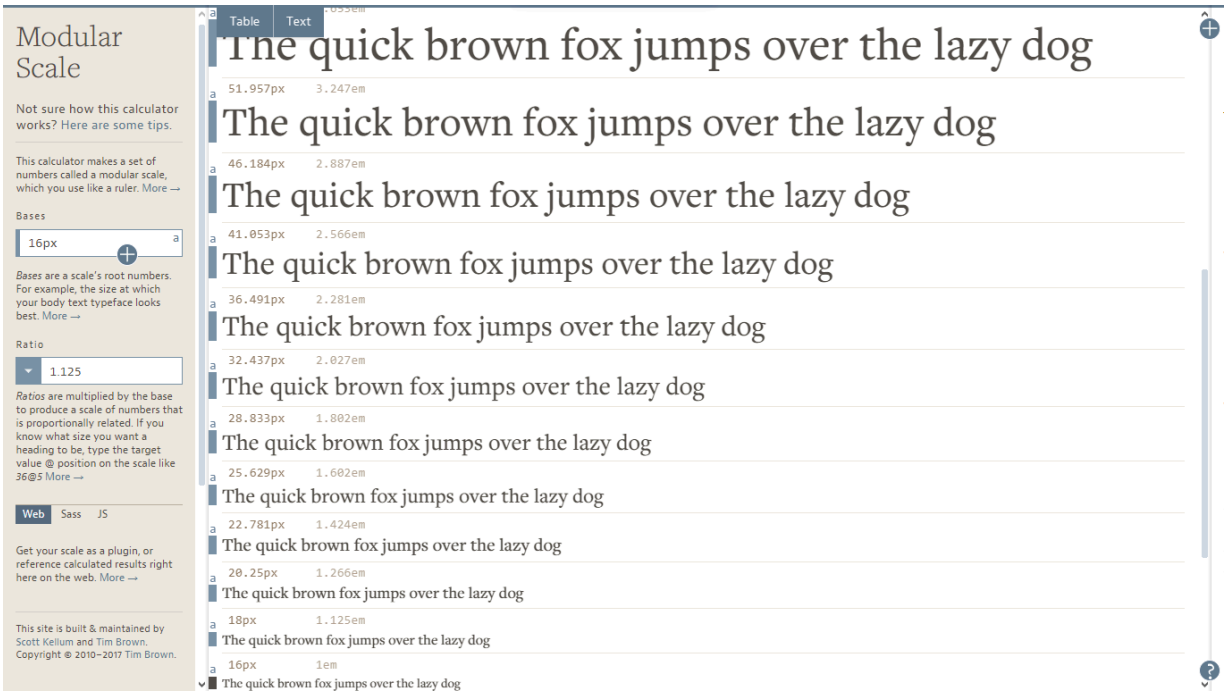


Figura 17.1: Página do site Modular Scale. À esquerda estão as definições de 16 pixels para o tamanho base do texto. Abaixo está a proporção de 1.125. À direita, estão as escalas textuais em diversos tamanhos, representadas pelo texto The quick brown fox jumps over the lazy dog.

Você vai se deparar com uma grande variedade de proporções: algumas geram diferenças sutis entre os tamanhos de texto, enquanto outras aumentam bruscamente essas variações. Surge então a dúvida: qual delas escolher? A verdade é que não existe uma proporção melhor do que a outra — tudo depende de compreender quais hierarquias precisam ser atendidas. A partir da escala selecionada, é necessário avaliar se ela realmente ajuda a diferenciar bem um estilo de texto do outro. Quanto maior o valor da proporção da escala, maiores serão as diferenças entre os tamanhos.

Pensando em um exemplo prático: temos o valor base de 16 pixels. A partir dele, subimos 5 escalas (17, 18, 19, 20 e 21) e descemos 2 escalas (15 e 14). Contando a base, temos 16px + 5 escalas para cima + 2 escalas para baixo = 8 escalas no total. Uma proporção da

escala modular que pode não ser tão adequada é a de valor 1.5, chamada *Perfect Fifth*. O maior estilo de texto ficaria com 121.5 pixels, impossibilitando o uso para qualquer frase, já que ocuparia muito espaço da página, podendo até quebrar palavras longas em linha.

Outra forma de reforçar a hierarquia textual, que certamente você conhece, é o uso dos pesos das fontes, como "regular", "medium", "bold" e outros. De fato, eles são muito utilizados para dar destaque a palavras ou frases, chamando a atenção do leitor. Mas também são comumente empregados para tornar títulos mais evidentes e impactantes.

Medidas relativas

No papel de designer, especialmente para quem cria interfaces digitais, é bastante difícil trabalhar com medidas que não sejam o pixel. Isso se deve muito às ferramentas que temos no mercado para trabalhar com protótipos, como é o caso do próprio Figma, que nos limita a atuar com apenas uma medida. Contudo, como vimos no capítulo anterior, temos uma série de outras unidades que nos permitem ter mais flexibilidade na interface.

Pixels não são unidades de medida ruins. Na verdade, são muito boas para estabelecer medidas exatas. Porém, quando levamos isso para o lado tipográfico, surgem pontos negativos, como a falta de adaptação quando os usuários aumentam o tamanho base nas configurações do navegador e a baixa flexibilidade para diferentes dispositivos. Por essa razão, a medida `rem` é mais utilizada para definir o tamanho dos textos.

Recapitulando: `rem` é uma unidade totalmente relativa, baseada no tamanho da fonte raiz do projeto, o que significa que o valor final de `1rem` sempre dependerá do tamanho definido no `<html>`.

Como já mencionei, os navegadores costumam utilizar 16 pixels como o tamanho padrão da fonte raiz; logo entendemos que `1rem` equivale a 16 pixels, `2rem` equivale a 32 pixels, e assim

sucessivamente. O uso desses valores relativos facilita a definição e a flexibilidade do tamanho dos textos, pois evita que uma escala permaneça com um tamanho desnecessariamente grande em um dispositivo de tela menor.

O conceito de escala modular é muito bem aplicável a esse contexto quando pensamos em uma adaptação flexível dos nossos textos de dispositivos grandes para pequenos e vice-versa.

Em vez de usarmos valores em pixels declaradamente, podemos simplesmente converter o tamanho de cada escala para a unidade `rem`. Por exemplo: se considerarmos que, em um ambiente *desktop*, o tamanho da fonte raiz foi definido em 16 pixels, e utilizarmos uma proporção de 1.2, obteremos a seguinte escala:

px	rem
27.648	1.728
23.04	1.44
19.2	1.2
16	1
13.333	0.833

Por ser uma unidade dinâmica, não precisamos alterar nenhum dos valores em `rem`. Basta atualizarmos o tamanho base (16 pixels) para os outros *breakpoints*. Se aplicarmos 14 pixels como o tamanho base para dispositivos móveis, mantendo a mesma proporção, temos os mesmos resultados em `rem`, mas conversões diferentes em pixels:

px	rem
24.192	1.728
20.16	1.44

px	rem
16.8	1.2
14	1
11.667	0.833

17.3 Altura de linha e espaçamento entre caracteres

O elemento tipográfico é um dos que mais possuem características a serem trabalhadas, e cada uma dessas propriedades pode se beneficiar significativamente das escalas. Legibilidade e leiturabilidade são aspectos que sempre queremos garantir para as pessoas que lerão o texto no produto, principalmente aqueles textos longos e corridos.

E uma dessas propriedades que também pode ajudar com a leiturabilidade das informações é a **altura de linha**, também conhecida como *leading*. A principal missão dessa propriedade é manter o conforto na leitura, criando espaçamentos que facilitem o entendimento e o acompanhamento do texto. No entanto, é preciso cuidado, pois um espaçamento mal planejado pode prejudicar a leitura.

É possível projetar as alturas de linha também utilizando os valores das escalas modulares, mas de forma muito mais simples, já que a aplicação da altura de linha se baseia totalmente no tamanho do seu texto. Por exemplo, se você estiver com um parágrafo cujo tamanho tem 16 pixels, e definir que a altura de linha será 1.5, o próprio

navegador fará essa multiplicação entre os dois valores, que resultará em 24 pixels.

Um ponto importante ao trabalhar com alturas de linha é testar diferentes aplicações, pois cada fonte é elaborada com uma altura-x diferente da outra. Isso pode resultar em espaçamentos maiores ou menores entre as linhas.

Uma recomendação é pensar em uma altura de linha específica para cada categoria de texto, como o *body text* e os *headings* mencionados. Isso ocorre porque, quanto maior o tamanho da fonte, maior deve ser a altura de linha. Se um título ocupar duas linhas ou mais, o espaçamento pode se tornar excessivo; ajustar a altura de linha nesse caso garante conforto na leitura, evitando respiros desnecessariamente longos.

ANDROID

O poder da IA do Google chega à nova série Samsung Galaxy S24

1.2 de altura de linha

17 Jan, 2024 · 4 minutos de leitura

Compartilhar

Hiroshi Lockheimer
Senior Vice President, Platforms & Ecosystems

ANDROID

O poder da IA do Google chega à nova série Samsung Galaxy S24

1.5 de altura de linha

17 Jan, 2024 · 4 minutos de leitura

Compartilhar

Hiroshi Lockheimer
Senior Vice President, Platforms & Ecosystems

Figura 17.2: Duas versões da altura de linha aplicada na publicação "O poder da IA do Google chega à nova série Samsung Galaxy S24". Acima está a versão com 1.2 de altura de linha; abaixo está a versão com 1.5.

Quando falamos de espaçamento em tipografia, a altura da linha é, sem dúvida, a mais recorrente, mas não a única. O **espaçamento entre caracteres**, também chamado de *tracking*, é uma propriedade frequentemente esquecida e pouco explorada por designers, podendo contribuir significativamente para a legibilidade quando combinada com as demais características discutidas anteriormente.

Se você notar, o ajuste das outras características sempre afeta o espaçamento entre os caracteres. Regular o tamanho ou o peso de um texto também aumenta ou diminui esse espaçamento. Esse cuidado é especialmente necessário ao trabalhar com textos muito grandes, normalmente *headings*, ou muito pequenos, que exercem funções de legenda.

Sobre esse assunto, um artigo publicado em agosto de 2015 pelo engenheiro front-end John D. Jameson apresentou, com exemplos, os cenários ideais para aplicar o espaçamento entre caracteres. Uma das aplicações envolve textos de cor clara sobre fundos escuros. Nesses casos, o texto pode aparentar ter um peso maior; portanto, aumentar o espaço entre os caracteres em interfaces escuras ou no modo noturno pode ajudar na leitura.

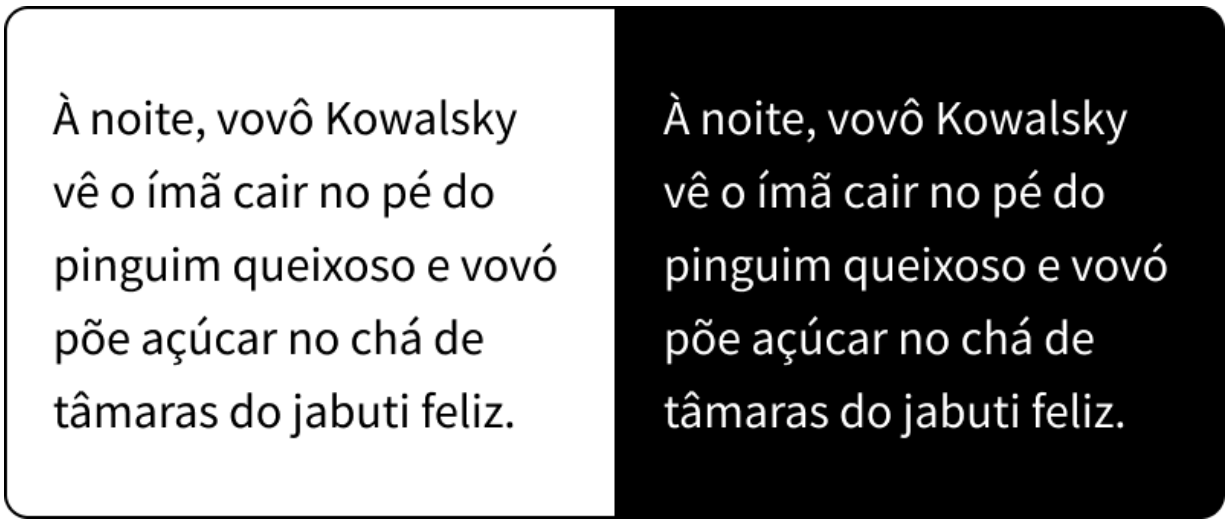


Figura 17.3: À esquerda está um texto de cor preta aplicado a um fundo branco, enquanto à direita está um texto branco aplicado a um fundo preto. O texto escrito é "À noite, vovô Kowalsky vê o ímã cair no pé do pinguim queixoso e vovó põe açúcar no chá de tâmaras do jabuti feliz".

Capítulo 18
Design tokens

No cenário complexo do design digital, especialmente na gestão de design systems, a busca pela consistência e eficiência é constante. À medida que o design evolui e os produtos digitais se expandem, surge a necessidade de uma abordagem mais organizada e dinâmica na gestão da estética e comportamento visual. É nesse contexto que os design tokens emergem como uma solução inteligente e eficaz.

Design tokens são como átomos visuais. São unidades fundamentais e semânticas de design que armazenam propriedades visuais, como cores, tipografia e espaçamentos, em uma forma reutilizável e padronizada. De certa forma, são parecidos com variáveis ao guardar um determinado valor, mas o que os diferencia é sua aplicação, já que os design tokens têm objetivos mais específicos e semânticos para a gestão de estilos visuais, enquanto as variáveis são mais genéricas, usadas para diferentes casos.

Cada token representa um único atributo visual e age como uma abstração, dissociando o estilo visual do código-fonte. Essa dissociação oferece flexibilidade e coesão ao permitir que alterações em um token afetem automaticamente todos os elementos visuais associados.

A implementação de design tokens traz uma série de benefícios. Primeiramente, eles oferecem uma maneira de gerenciar a consistência visual em todos os elementos do design system, garantindo uma experiência coesa para o usuário. Além disso, promovem a reusabilidade e facilitam a manutenção, permitindo que as pessoas designers e desenvolvedoras usem e ajustem rapidamente os tokens conforme necessário.

Esse é um dos grandes benefícios que fazem com que os design systems se tornem produtos que asseguram resultados ágeis, pois

não apenas mantêm a integridade visual, como também aceleram o processo de design e desenvolvimento.

18.1 Formato

O formato mais comum de se trabalhar com design tokens é usando JSON (JavaScript Object Notation), seguindo uma estrutura de "nome", "valor" e "metadados":

```
{
  "color": {
    "brand": {
      "primary": {
        "value": "#1E90FF",
        "type": "color",
        "description": "Cor principal da marca"
      }
    }
  }
}
```

O nome que damos a um design token indica sua categoria e atribuições, como `color.brand.primary`. O valor corresponde à informação armazenada por aquele token, como `#1E90FF`, e que será aplicada no projeto. O metadado é a informação que ajuda a dar o contexto de uso — que podem ser vários, como "color", "typography", "border", "spacing" etc. Como a lista é grande, sugiro consultar a página *Design Tokens Format Module* (https://www.designtokens.org/tr/drafts/format/) da Design Tokens Community Group.

A vantagem de trabalhar com esse formato em JSON reside tanto na simplicidade de leitura e compreensão das informações, quanto por ser uma linguagem comum entre diferentes plataformas.

Esse modelo facilita a comunicação entre design e código, permitindo que plataformas e plugins ofereçam aos designers a possibilidade de gerenciar os design tokens de forma visual e exportá-los em um ou mais arquivos.

18.2 Níveis

A proposta de criar design tokens é pensar em sua escalabilidade. A pequena comparação que fiz com variáveis torna mais simples entender o início da sua construção, que nada mais é do que um dos nossos estilos visuais, como um código de cor ou um tamanho de texto, que passa a ter um nome para facilitar a utilização, ajustes e reconhecimento.

O processo de construção e evolução é uma jornada que parte do abstrato para o concreto, transformando valores fundamentais em elementos que terão casos de uso específicos na interface. Imagine começarmos com um tom azul claro, de hexadecimal #1E90FF. Inicialmente, esse valor pode ser encapsulado em um **primitive token**, que seria o primeiro nível de classificação e que representa, como essência, uma aplicação global. Com isso, podemos começar a chamá-lo de `blue-400` (exemplo).

Apesar de sua identificação ter ficado mais prática, não é possível determinar onde o token blue-400 pode ser utilizado. É para isso que os primitive tokens são refinados e se tornam **semantic tokens**, trazendo uma finalidade de aplicação mais específica. Por exemplo, se o design token blue-400 deve ser utilizado para *call to actions* (chamadas de ação), então criamos o token `cta-background`.

Pode parecer que não há mais alternativas, mas a parte mais interessante dos design tokens é como eles podem ser cada vez mais afunilados. Ao analisarmos a quantidade de componentes que podem ser um *call to action*, como um botão ou um link,

percebemos a necessidade de adicionar mais uma camada de design token. Nesse caso, os semantic tokens são refinados para **component tokens**, resultando na criação do "button-cta-background".

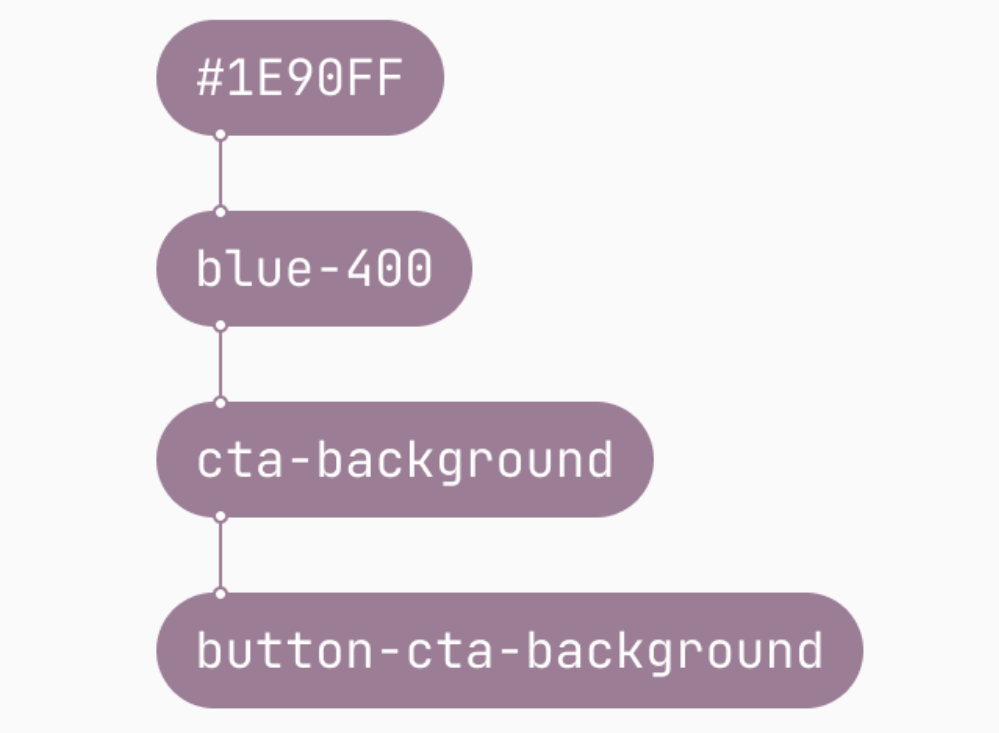


Figura 18.1: Quatro itens aparecem ligados, um abaixo do outro. No primeiro temos o código hexadecimal #1E90FF. O segundo está com o design token blue-400. No terceiro temos cta-background. O quarto e último está com button-cta-background.

Para resumir, podemos trabalhar com três níveis de design tokens:

- **Nível 1 - Basic tokens:** usados para definir a aplicação global de um determinado valor.
- **Nível 2 - Semantic tokens:** usados para refinar e permitir um uso mais específico aos primitive tokens.
- **Nível 3 - Component tokens:** usados para refinar e possibilitar um uso mais específico aos semantic tokens.

Esse é um dos cenários mais comuns e utilizados para a definição e refinamento de design tokens, com apenas três níveis. Mas considere que nada é escrito em pedras, e você pode se deparar com empresas que trabalham com mais níveis ou até mesmo usam nomenclaturas diferentes.

O próprio Spectrum, design system da Adobe, utiliza desse modelo de três níveis para a criação de seus design tokens:

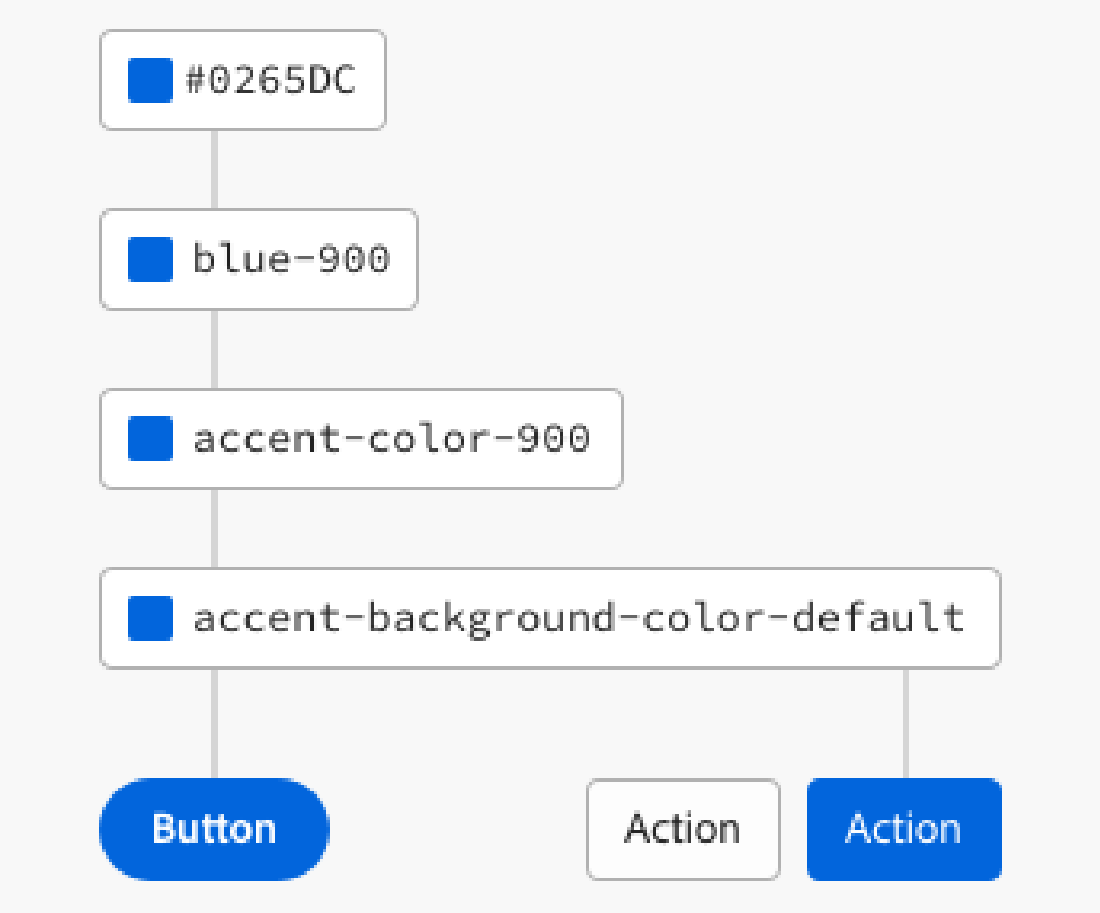


Figura 18.2: Quatro blocos de evolução de uma cor para um design token. No primeiro está o código hexadecimal #0265DC. Abaixo, temos um design token chamado blue-900. Abaixo dele, foi formado o accent-color-900. E, por fim, o último token chamado de accent-background-color-default. Ele está ligado em dois botões diferentes que possuem cor azul no fundo. Imagem retirada da página do Spectrum (https://spectrum.adobe.com/page/design-tokens/).

Se esta for a primeira vez que você lê sobre design tokens e está conhecendo sua aplicação, não se assuste se o conteúdo pareceu

complexo, dada a abrangência de informações que podem ser levantadas e afuniladas. Acredito que todo profissional que atua ou já atuou com isso encontrou desafios por alguns dias ou semanas, pois a construção de uma biblioteca de design tokens não é um trabalho de um único dia.

18.3 Nomenclatura

Não existe um padrão que indique como deve ser a nomenclatura de um design token. Podemos observar isso nos próprios design systems que eu já trouxe ao longo deste livro como referência.

Contudo, a forma que recomendo para elaborar a convenção de nomenclatura é pensar nos grupos e subgrupos de cada parte. Vamos entender melhor com o seguinte exemplo:

```
color-button-primary-background-hover
```

No caso deste token, podemos entender duas coisas. A primeira é que ele é um token de componente, visto que no meio do nome temos "button". A segunda é que ele está relacionado à cor de fundo do botão em estado hover da variante primária.

Essa estrutura combina uma série de atributos em ordem. Se eles forem lidos da esquerda para a direita, podemos entender que cada termo funciona como **pai** do termo seguinte. Ou seja, "color" é pai de "button", que por sua vez é pai de "primary" e assim por diante. No caso do exemplo, estamos tratando de:

- Categoria.
- Componente.
- Variante.
- Propriedade.
- Estado.

Este é apenas um exemplo isolado, pensado a nível de componente. É importante trabalhar os tokens de forma faseada, iniciando sempre pelos primitive tokens, como vimos anteriormente. É a partir do básico, listando cada parte que estrutura um token, que ampliamos os níveis de categorias e subcategorias. Para ilustrar, vamos analisar um caso real com o modelo do Pajamas Design System, do GitLab.

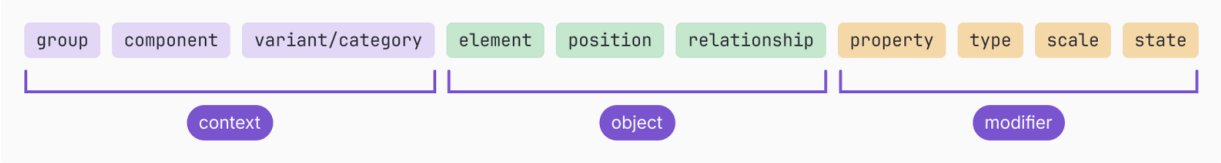


Figura 18.3: Diagrama que mostra a estrutura de nomeação de design tokens e componentes. À esquerda, 'context' agrupa: group, component e variant/category. No centro, 'object' contém: element, position e relationship. À direita, 'modifier' inclui: property, type, scale e state.

O GitLab, mesmo utilizando nomenclaturas diferentes, também trabalha com os mesmos três níveis de design tokens. Ou seja, sua estrutura escala desde o encapsulamento de valores até a contextualização em componentes. Dentro dessa estrutura, há três grandes áreas, cada uma com uma série de categorias diferentes:

- **Contexto:** indica em qual cenário aquele design token estará encaixado;
- **Objeto:** indica em qual elemento aquele design estará presente;
- **Modificador:** indica a intenção de uso do elemento.

Essa tratativa das áreas, por mais "bonita" que seja, não é necessariamente importante de definir. O foco deve estar em cada um dos grupos que formam a nomenclatura completa, e nos atributos de cada um deles. Para chegar a uma versão completa, podemos nos basear nos seguintes tópicos:

1: Padrão de escrita

Antes de qualquer coisa, escolha como os nomes dos design tokens serão escritos, de forma a manter a consistência. Alguns desses

padrões são:

- `kebab-case` : modelo mais usado, formado por letras minúsculas e palavras separadas por hífen (exemplo: `font-size-sm`).
- `snake-case` : semelhante ao anterior, mas usando *underline* em vez de hífen (exemplo: `font_size_sm`).
- `camelCase` : modelo em que todas as palavras ficam juntas, com a primeira letra de cada palavra em maiúscula, com exceção da primeira palavra (exemplo: `fontSizeSm`).

2: Estrutura hierárquica

Design tokens normalmente seguem uma ordem hierárquica, em que o item à direita funciona como um 'subgrupo' do item à esquerda. Nem todos os design systems precisam do mesmo nível de detalhamento que o GitLab, mas você pode se basear nele. Um bom ponto de partida pode considerar:

`categoria-variante-escala` sendo uma escala menor, para tokens primitivos. `categoria-variante-propriedade-escala` uma escala maior, com definição de propriedade para tokens semânticos. `categoria-componente-variante-propriedade-escala-estado` é um modelo completo, atendendo também à necessidade de componentes.

De nada adianta apenas pensar na estrutura se não definirmos o que fará parte de cada uma delas. Podemos utilizar o modelo mais completo para identificar os atributos que o compõem, de forma que a hierarquia nos ajude a formar um design token completo:

- Categoria: `color` , `border` , `spacing` **...**
- Componente: `alert` , `button` , `card` **...**
- Variante: `primary` , `secondary` , `danger` **...**
- Propriedade: `background` , `size` , `line-height` **...**
- Escala: 100, 200, 300, `sm` , `md` , `lg` **...**
- Estado: `hover` , `pressed` , `focus` **...**

3. Escala consistente

Muitos dos design tokens com os quais trabalhamos, como cores, espaçamentos e bordas, possuem vários níveis ou escalas que seguem uma lógica de aumento e diminuição de valores. Usamos escalas porque elas criam previsibilidade e consistência dentro do sistema, permitindo que designers e desenvolvedores entendam como os valores se relacionam e escolham a opção mais adequada sem precisar criar algo novo. Existem diferentes modelos de escalas que podem ser adotados nos design tokens, considerando que os mais comuns são:

O **modelo progressivo** costuma escalar de 100 em 100 (`color-blue-100` , `color-blue-200` ...) e, com menos frequência, também de 1 em 1 e 10 em 10. É muito usado em cores e sombras para identificar o peso ou intensidade.

O **modelo t-shirt** possui um formato diferente, usando as siglas comuns em camisetas no padrão estadunidense, como `sm` (*small*), `md` (*medium*) e `lg` (*large*). Essa convenção costuma aparecer em atributos de tipografia, como `font-size` e `line-height` , mas também em elementos, como `border-radius` . É utilizada principalmente para indicar o tamanho ou nível de um token, mas pode dificultar a compreensão do valor, especialmente quando a sigla é aplicada em diferentes categorias, variantes e propriedades.

Além dos modelos anteriores, é possível usar formatos mais descritivos, como `compact` e `comfortable` para detalhar tamanhos específicos de componentes, ou termos como `light` e `dark` , `lighter` e `darker` , `lightest` e `darkest` para tratar níveis claros e escuros de cores.

4. Nomenclatura clara

Já recebi perguntas sobre se um token grande poderia ser algo negativo, e não há nada que indique que isso seja prejudicial ao sistema. Nesse ponto, devemos nos concentrar na clareza da comunicação, independentemente do tamanho do token.

Evite abreviações ou termos genéricos que dificultem a compreensão do propósito de um design token. Por exemplo, um `c-bg-prm`, por mais rápido que seja de digitar, pode confundir seu propósito, ao contrário de `color-background-primary`.

Parte 5 - Componentes e microinterações

Os componentes são elementos essenciais de um design system, responsáveis por padronizar a interface e otimizar a criação de produtos digitais. Eles podem variar de partes simples, como botões e campos de entrada, até estruturas mais elaboradas, como tabelas e modais. Seu principal objetivo é garantir uniformidade visual e funcional, facilitando a colaboração entre designers e desenvolvedores.

Para que cumpram esse papel de forma eficiente, os componentes devem ser bem documentados, contemplando variações, estados e diretrizes de uso. Isso permite que sejam aplicados corretamente em diferentes cenários, evitando inconsistências e retrabalho. Quando bem estruturado, o sistema de componentes contribui diretamente para a escalabilidade e manutenção do design system.

Nesta quinta parte do livro, você lerá sobre:

- O que são componentes.
- Os tipos de componentes de um design system.
- Os estados e microinterações.

Capítulo 19
Componentes

Os componentes são elementos reutilizáveis e independentes que compõem a interface de um produto digital, cada um encapsulando uma funcionalidade própria. Eles são projetados para serem modulares e consistentes, permitindo que sejam facilmente combinados e reutilizados em diferentes partes do projeto, melhorando a eficiência do desenvolvimento, a consistência visual e a manutenção do código.

Imagine o cenário de construção de um site ou aplicativo, em que há a necessidade de vários botões, campos de texto, barras de navegação e cards. Em vez de designers e desenvolvedores sempre precisarem criar novos elementos do zero, o que seria muito trabalhoso e propenso a erros, o design system fornecerá todos os componentes de que os colaboradores precisarão.

Por fazerem parte de uma biblioteca em que o propósito é contemplar os mais diversos cenários para todos os produtos, esses componentes já virão com todas as definições prontas. Isso inclui estilos, microinterações, acessibilidade, padrões de design e seus respectivos casos de uso.

Essas definições tornam mais eficaz o processo de tomada de decisão, uma vez que, ao conhecer os cenários, sua inclusão e composição se tornam mais fáceis e agilizam a produção. São peças fundamentais para a composição das páginas do produto, pois permitem a criação de elementos de interface de forma eficiente, consistente e reutilizável.

Além disso, os componentes possuem uma similaridade com os design tokens quando falamos da facilidade de manutenção e escalabilidade. Por exemplo, se você precisar atualizar o estilo de um botão, esse ajuste será feito diretamente no componente do botão, tanto do ponto de vista do design quanto da programação. Dessa forma, todas as instâncias desse botão nos projetos que usam o design system serão automaticamente atualizadas.

Considerando os benefícios, para começar a trabalhar com os componentes é necessário dividi-los em duas categorias principais: os ***core components*** e os ***team components***. Essa classificação permite uma organização eficiente dos elementos de design, atendendo tanto às necessidades gerais do sistema quanto às especificidades de cada equipe ou funcionalidade.

Core Components

Os *core components* são os principais componentes do design system, ou seja, aqueles que todos os designers e desenvolvedores têm acesso ao utilizarem o design system na construção de um produto, e que são constantemente escalonados pelo time responsável.

Essas são as interfaces que atendem às demandas em comum dos produtos, sem pensar em um determinado cenário. Dentre os mais comuns, posso citar o *alert*, *button*, *card*, *select* e *text field*.

Considere, então, um design system que alimenta quatro produtos e que possui, ao todo, 25 componentes, independentemente do caso de uso. É importante que todos esses componentes sejam aplicáveis a todos os produtos, evitando casos específicos que ocasionem a manutenção de itens na biblioteca que sirvam apenas para um produto.

Isso não significa que novos componentes não possam surgir. O design system deve estar aberto a novas necessidades gerais. No entanto, é extremamente comum que determinados produtos demandem interfaces complexas que o design system não contempla, justamente por se tratarem de cenários específicos. É nesse momento que entram os *team components*.

Team Components

Os *team components* são desenvolvidos pelos designers e desenvolvedores dos produtos que consomem o design system, com o objetivo de atender a necessidades específicas de suas interfaces. Eles são construídos a partir dos *core components*, herdando integralmente a base visual e funcional, mas incorporando regras de negócio, variações ou combinações que fazem sentido apenas para o produto em questão.

Um aspecto interessante é que, sempre que viável, os *team components* podem evoluir e ser promovidos a *core components*, passando a integrar a biblioteca principal do design system. Para

isso, a equipe responsável deve avaliar cuidadosamente se o componente atende a uma necessidade mais ampla e se possui potencial de reutilização em outros produtos da empresa.

A promoção de *team components* para *core components* não é obrigatória, e eles não precisam ser encarados necessariamente como uma etapa de evolução rumo à biblioteca principal. Ainda assim, é importante alinhar expectativas com os responsáveis pelo *team component*, deixando claro que não devem aguardar sua promoção ou mesmo seu desenvolvimento pela equipe do design system, já que essa é uma responsabilidade integral do time do produto.

Capítulo 20
Propriedades e customizações

Um dos ganhos de se ter um design system, como você já sabe, é evitar problemas relacionados a inconsistências visuais. Por exemplo, ao trabalhar com botões, definimos quais design tokens serão utilizados para definir cores, bordas, espaçamentos e outros estilos visuais que impedem que o botão fuja dessas propriedades.

Contudo, o contexto pode demandar que a estrutura desse componente mude para se adaptar corretamente, e é para isso que trabalhamos com propriedades. As propriedades são atributos que permitem customizar um componente de diferentes formas, mas que continuem seguindo os padrões já estabelecidos pelo design system. Entenda algumas das mais comuns a seguir:

Variante

Você já deve ter notado que um mesmo componente pode ter diferentes aplicações visuais, como é um caso frequente nos botões. Um botão pode aparecer em várias formas: em destaque, com fundo colorido, outro mais discreto, com fundo transparente e até mesmo mais simples, apenas com texto. Tudo isso em um único componente: é o que chamamos de variante.

As variantes são uma forma de definir e controlar essas diferentes versões de um componente, mantendo a consistência do visual enquanto oferece flexibilidade para diferentes cenários. Por ser um único componente, é possível que todas as diferentes variações, mesmo que possuam estilos diferentes, tenham estruturas idênticas, como arredondamento da borda, paddings e estilos de texto.

Por exemplo, imagine um campo para publicação de postagem de blog. Só nessa página, podemos encontrar pelo menos dois botões, sendo um para publicar e outro para salvar como rascunho. No

geral, são o mesmo componente, mas, devido às diferentes hierarquias, cada um terá uma variante aplicada.

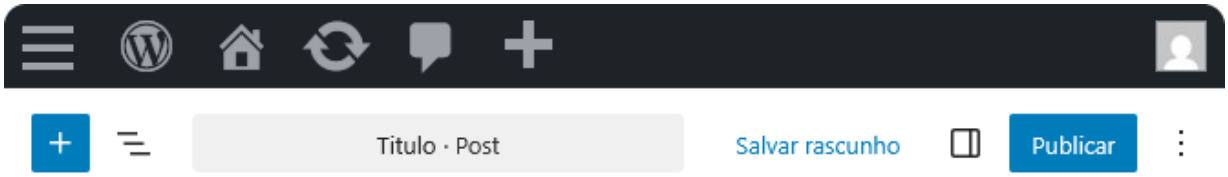


Figura 20.1: Página de publicação de postagem do WordPress. No topo, está um botão com fundo azul chamado Publicar, e ao lado esquerdo, um botão com fundo transparente e texto cinza chamado Salvar rascunho.

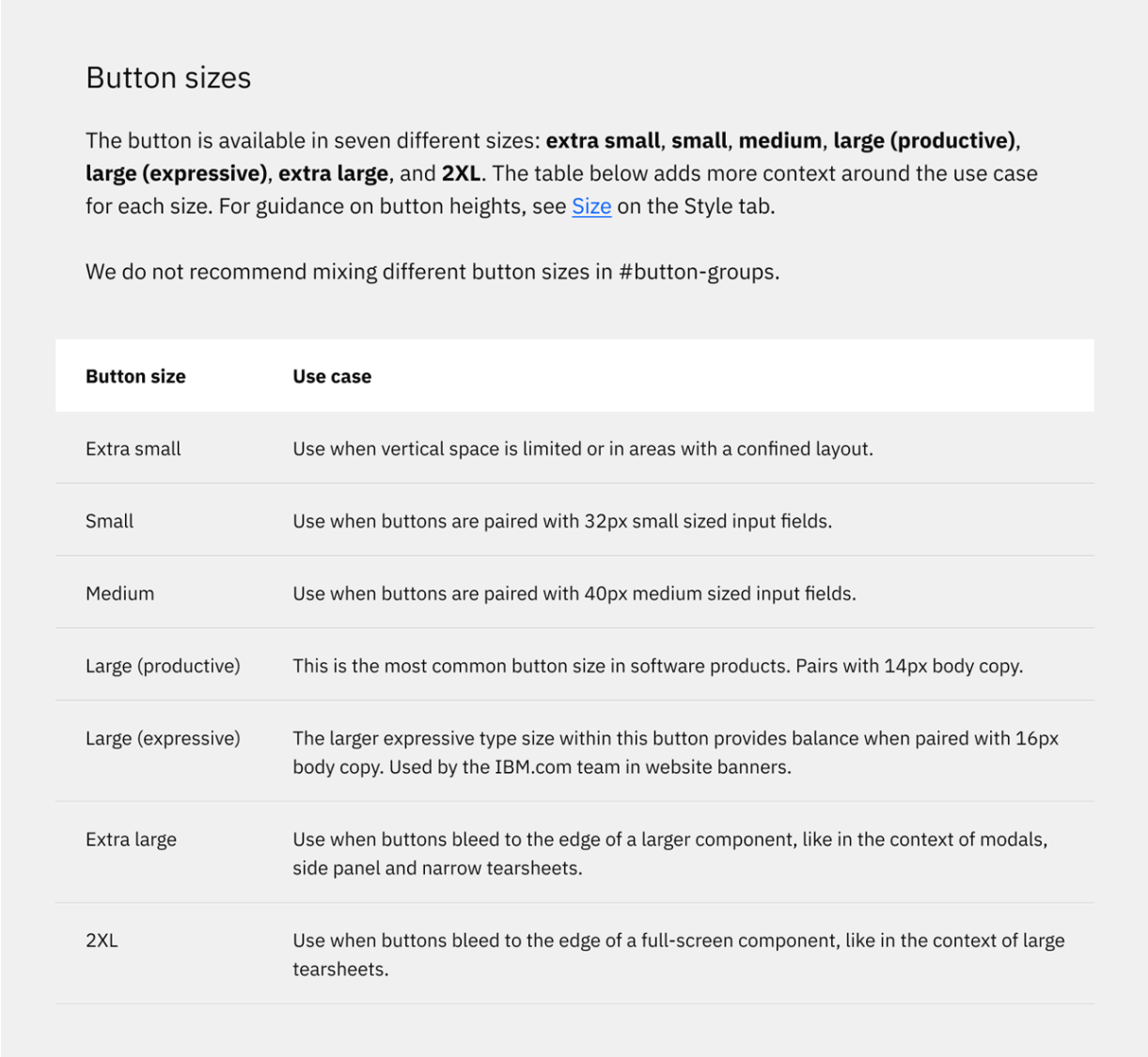
Tamanho

Você já deve ter notado que, dependendo de onde o componente está, o tamanho dele precisa ser ajustado para garantir que se encaixe bem no layout, esteja alinhado corretamente com outros elementos e cumpra sua função de forma eficiente. Imagine o quão estranho seria um botão pequeno, com 28 pixels de altura, alinhado à direita de um campo de texto com 48 pixels de altura. Casos como esse exigem que trabalhemos com propriedades de tamanho.

A propriedade de tamanho se refere aos ajustes de dimensões dos componentes, podendo variar tanto para tamanho pequeno quanto grande. Isso pode incluir os paddings, gaps, tamanho de texto, espessura da borda, arredondamento de borda e qualquer outra propriedade visual que realce ou suavize os detalhes.

Essa é uma propriedade muito bem aproveitada junto das variantes, pois mesmo que elas mudem estilos para definir hierarquias ou cenários diferentes, ainda podem ter seu tamanho modificado.

Um exemplo real deste tipo de aplicação está no botão do Carbon Design System, que trabalha com os tamanhos extra small, small, medium, large (productive), large (expressive), extra large e 2XL. Cada um desses tamanhos possui um caso de uso, considerando cenários como usos convencionais, espaços limitados, tamanhos de texto específicos e até mesmo em união com campos de formulário.



Button sizes

The button is available in seven different sizes: **extra small**, **small**, **medium**, **large (productive)**, **large (expressive)**, **extra large**, and **2XL**. The table below adds more context around the use case for each size. For guidance on button heights, see Size on the Style tab.

We do not recommend mixing different button sizes in #button-groups.

Button size	Use case
Extra small	Use when vertical space is limited or in areas with a confined layout.
Small	Use when buttons are paired with 32px small sized input fields.
Medium	Use when buttons are paired with 40px medium sized input fields.
Large (productive)	This is the most common button size in software products. Pairs with 14px body copy.
Large (expressive)	The larger expressive type size within this button provides balance when paired with 16px body copy. Used by the IBM.com team in website banners.
Extra large	Use when buttons bleed to the edge of a larger component, like in the context of modals, side panel and narrow tearsheets.
2XL	Use when buttons bleed to the edge of a full-screen component, like in the context of large tearsheets.

Figura 20.2: Seção de tamanhos de botão na documentação do Carbon Design System. No topo está o título "Button sizes", e abaixo está uma tabela de duas colunas, sendo que à esquerda estão as variantes small, medium, large productive, large expressive e extra large. À direita está a descrição do caso de uso de cada uma das variantes.

As especificações de tamanho também podem ser aproveitadas para definir larguras mínimas e máximas, como em *drawers*, modais e *toasts*. Essas limitações ajudam a evitar que esses componentes tenham espaços vagos em casos de pouca informação ou até mesmo que fiquem muito apertados para cenários em que há uma quantidade considerável de informações para exibir.

Exibir e ocultar ícones

Como você bem sabe, ícones são ótimos recursos visuais para complementar os textos, pois ajudam a expressar o teor da mensagem mais rapidamente. Vemos ícones aparecendo em botões, menus, e até em campos de texto, mas de forma variada. Essa flexibilidade é importante para adaptar o componente ao contexto e ao espaço disponível na interface.

A propriedade de exibir ou ocultar ícones serve para controlar o padrão de comunicação de um grupo de componentes. Como um dos objetivos de um design system é manter a consistência, quando temos um grupo de componentes, incluindo diferentes botões reunidos em uma página, ao utilizar o ícone em um deles, o ideal é que todos sigam o mesmo padrão.

Contudo, além de serem usados apenas para leitura, podem também ser habilitados para executar alguma ação, como é frequente nos campos de senha com o ícone de olho para exibir a senha ou nos campos de busca com a lupa para seguir com a pesquisa.

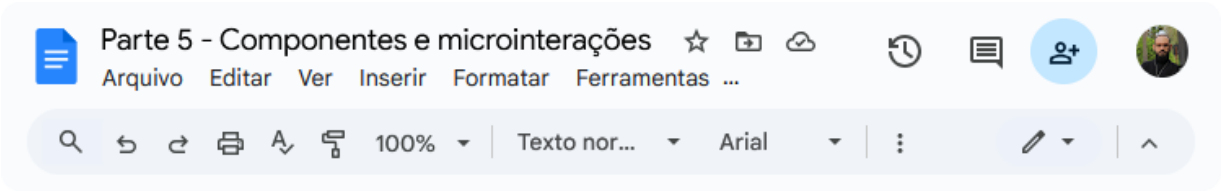


Figura 20.3: Barra superior do Google Docs com o título do documento "Parte 5 - Componentes e microinterações", mostrando os menus Arquivo, Editar, Ver, Inserir etc., ícones de edição e colaboração, além da foto de perfil do usuário no canto superior direito.

Capítulo 21
Estados

Os estados dos componentes referem-se às diferentes condições e aparências que um componente pode assumir ao interagir com o usuário.

Esses estados têm um papel importante em proporcionar uma experiência de usuário dinâmica e informativa, comunicando visualmente mudanças ou *feedback* no comportamento dos elementos de interface. Tratam-se de variantes específicas que refletem o estado atual de um componente, influenciadas pelas ações ou dados fornecidos pelo usuário ou sistema.

Quando se trata de interação com componentes, existem dois tipos de estados: os que possuem e os que não possuem microinterações. Não há obrigação em aplicar todos, mas, de forma geral, podemos citar os seguintes:

- Hover
- Pressed
- Focus
- Activated ou Selected
- Draggable
- Loading
- Error
- Read-only
- Disabled

Hover

O estado *hover* é fundamental para os componentes, pois proporciona *feedback* visual durante as interações do usuário. Quando o cursor do mouse se posiciona sobre um componente, como botões ou links, o estado entra em ação, alterando a aparência

para indicar que a área está interativa. Isso ajuda a criar uma experiência mais dinâmica, fornecendo retorno imediato sobre os elementos clicáveis.

Esse retorno visual e sutil, geralmente uma variação na cor, sombra ou estilo do componente, ajuda o usuário a entender as áreas navegáveis da interface. Sua utilização é valiosa para indicar que é possível interagir com o elemento e que uma ação poderá ser executada.

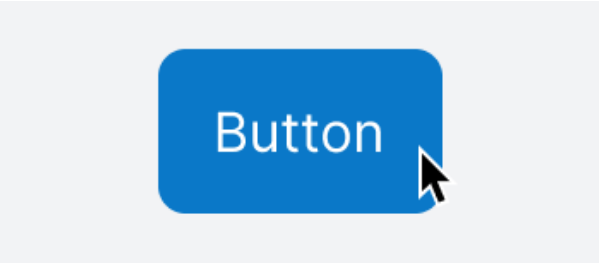


Figura 21.1: Um botão de cor azul, com o texto "Button" de cor branca. Sobre o botão, há um cursor no seu estado natural.

Pressed ou Active

Quando um usuário pressiona e mantém pressionado (com o mouse) um elemento interativo, como um botão, o estado *pressed* entra em cena, proporcionando um *feedback* imediato para indicar que a ação foi reconhecida. Isso cria uma resposta tátil que reforça a interação da pessoa usuária.

Visualmente, o estado *pressed* é frequentemente representado por mudanças na sombra, cor ou posição do componente, não sendo muito diferente das formas de alteração visual do estado *hover*. É interessante configurar esse estado, pois ele fornece confirmação instantânea, tornando a experiência de navegação mais envolvente e intuitiva.

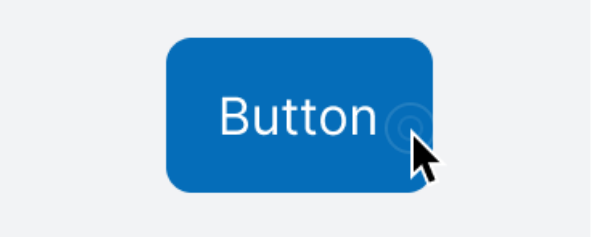


Figura 21.2: Um botão de cor azul escuro, com o texto "Button" de cor branca. Sobre o botão, há um cursor no seu estado natural, e na ponta desse cursor há um efeito de "ondas", indicando que houve ação de clique.

Focus

O estado focus é um dos mais importantes dessa lista, principalmente se consideramos navegação com teclado. Elementos interativos, como botões e campos de texto, entram em estado de foco logo após um clique ou após serem selecionados usando a tecla *Tab*. Em elementos apenas pressionáveis (*accordions*, botões, *checkboxes*...), quando estão nesse estágio, a interação pode continuar usando a tecla *Enter* ou a barra de espaço.

É frequentemente representado visualmente por uma borda ao redor do elemento, modelo convencional dos navegadores, mas que pode ser personalizado de diferentes formas. O mais importante para esse estado é que o usuário saiba qual elemento está em foco, facilitando sua localização na página.

Figura 21.3: Um botão azul com o texto "Button" na cor branca. Ao redor desse botão, há uma borda fina de cor azul, posicionada a 2 pixels de distância do elemento.

Activated ou Selected

Esses são estados que aparecem em componentes com modelos de interação diferentes, mas que seguem o mesmo sentido: demonstrar que algo foi ativado ou selecionado.

A aplicação desse estado normalmente é encontrada em elementos que foram escolhidos ou marcados pelo usuário, indicando sua seleção ativa. Vemos isso constantemente em alguns componentes de formulário, como *checkbox* e *radio button*, mas também em componentes de navegação, como *headers* e *navbars*, mais precisamente nos menus que indicam em qual página o usuário está.

Componentes com esse estado sempre terão o dobro de variações, pois necessitam de versões "ativada" e "não ativada". Isso significa que ambas as variações também podem apresentar os outros três estados de interação mencionados anteriormente.

Figura 21.4: Dois checkboxes lado a lado. Ambos possuem cantos levemente arredondados. O checkbox da direita não está habilitado e mostra apenas uma borda de cor cinza, sem fundo. Já o da direita está habilitado, mostrando um fundo azul, sem borda e um ícone de check na cor branca dentro.

Draggable

O estado *draggable* é associado a elementos que podem ser arrastados pelo usuário, geralmente usado em interfaces que suportam operações de arrastar e soltar. Essa capacidade de arrastar elementos adiciona uma camada interativa à interface, permitindo que os usuários reorganizem, movam ou interajam com conteúdo visual de maneira mais dinâmica.

Para componentes com esse estado, é comum que haja uma indicação visual no elemento que aparece apenas ao ser arrastado, muitas vezes associada a sistemas de elevação para criar o efeito de flutuação. Se o componente permitir apenas movimentação, sem outras interações embutidas, considere utilizar o cursor *grab*.

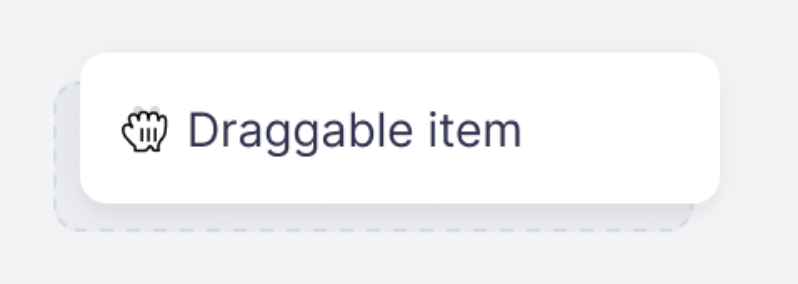


Figura 21.5: Um card com o texto "Draggable item" sendo arrastado com o cursor. O cursor está apresentado com uma mão fechada. O card é branco, com o texto de cor cinza escuro. O card possui uma sombra sutil e logo abaixo é mostrado uma área pontilhada que indica ser a área em que o card pode ser solto.

Loading

O estado *loading* é usado em componentes que indicam visualmente uma operação em andamento, ajudando o usuário a entender que algo está sendo carregado ou processado logo após o clique. Apenas uma pequena parcela dos componentes possui este estado, sendo mais comum em elementos específicos de carregamento, como barras de progresso, embora outros, como botões, também possam utilizá-lo.

O design system pode definir atributos visuais específicos para o estado *loading*, como animações de carregamento, ícones giratórios ou sobreposições semitransparentes. Esses indicadores visuais não apenas informam que a aplicação está ocupada, mas também fornecem uma representação gráfica intuitiva do progresso da operação.

Figura 21.6: Um card em estado de carregamento, usando "Skeleton loader" na imagem e texto, ocultando o conteúdo com blocos cinzas enquanto não é carregado.

Error

O estado *error*, assim como *loading* e *draggable*, não está presente em todos os componentes, sendo específico para cenários de falha ou erro durante uma operação. Seu uso é primordial para que o usuário identifique que algum problema ocorreu após realizar uma ação, fornecendo o *feedback* ideal.

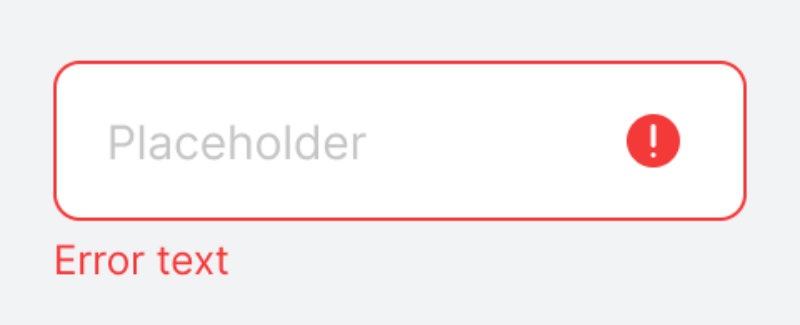


Figura 21.7: Campo de texto no estado de erro. Dentro do campo está o texto "Placeholder" alinhado à esquerda. Alinhado à direita está um ícone de exclamação na cor vermelha. A borda do campo também é vermelha, e abaixo do campo há um texto escrito "texto de erro", compartilhando do mesmo tom de vermelho.

Os componentes de formulário são os mais comuns que usam esse estado, já que usuários podem enviar formulários com informações incorretas, precisando de um retorno imediato.

Read-only

O *read-only* indica que o componente é apenas de leitura, permitindo somente a visualização. É um estado extremamente comum de se encontrar no componente *text field*, principalmente em sistemas que permitem visualizar um formulário com informações antes de ativar a edição dos campos.

O formato visual do componente escurece levemente a área em que a informação está escrita, para que o usuário consiga diferenciar do estado inicial.

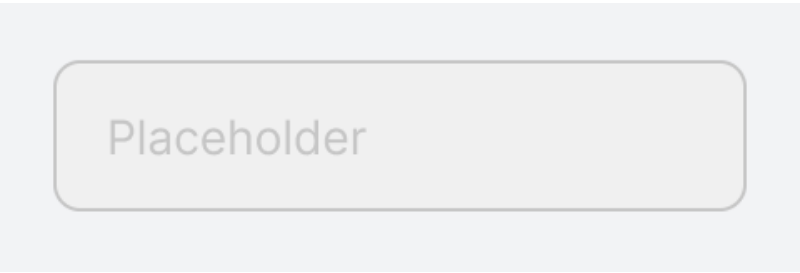


Figura 21.8: Campo de texto no estado de apenas leitura. A borda é de cor cinza, e o fundo num tom de cinza mais claro. O texto "placeholder" está dentro do campo.

Disabled

Se existem componentes que podem estar habilitados para uma interação, também devemos considerar que há a possibilidade desses mesmos componentes, em um determinado momento e cenário, estarem desabilitados, impedindo que qualquer função seja executada.

Nem todos os componentes requerem um estado desabilitado; somente aqueles que podem ou devem bloquear qualquer interação em algum momento. Os componentes de formulário são os mais comuns, especialmente quando o elemento não se aplica ao contexto ou está temporariamente indisponível.

As características visuais do estado *disabled* exigem uma diferença clara em relação ao estado padrão. Ele pode se assemelhar ao estado *read-only*, descrito anteriormente, com cor de fundo mais opaca e textos acinzentados, removendo qualquer detalhe que indique possibilidade de interação.

Figura 21.9: Campo de texto no estado desabilitado. A borda e o texto escrito "Placeholder" estão com o mesmo tom claro de cinza, enquanto o fundo do campo é branco.

Parte 6 - Boas práticas de documentação

A documentação de um design system é um guia abrangente que descreve os componentes, padrões e diretrizes de design utilizados para criar e manter a interface do usuário de forma consistente e eficiente. Ela serve como uma referência central para designers, desenvolvedores e outros membros da equipe, garantindo que todos sigam as mesmas normas visuais e funcionais, resultando em uma experiência de usuário coesa e harmoniosa.

Nesta sexta parte do livro, você lerá sobre:

- A estrutura ideal para a documentação de um design system.
- Como dividir e organizar o início da documentação para profissionais diferentes.
- Melhores práticas para documentar os patterns do sistema.

Capítulo 22
Iniciando a documentação

Após todas as etapas de planejamento, dinâmicas, definição de design tokens e desenvolvimento de componentes, chegamos à documentação — uma das partes mais importantes do design system, pois é ela que orientará os colaboradores sobre como utilizá-lo corretamente.

Não vou mentir para você. Essa pode não ser uma das partes mais divertidas da construção de um design system. As documentações podem assumir proporções enormes, nas quais até mesmo o componente mais simples pode resultar em mais de 10 mil caracteres e uma série de mídias inseridas.

De forma direta e sucinta, a documentação de um design system serve como um manual de teorias e práticas, não só mostrando tudo

o que o design system possui, mas também como utilizá-lo. A principal audiência da documentação é, sem dúvida, a equipe de design e desenvolvimento que consome o design system.

Para designers, o principal ganho da documentação será nos direcionamentos previamente elaborados para a aplicação dos estilos e componentes nos produtos, além dos padrões de uso e definição de conteúdo. Para programadores, o ganho será nas orientações de implementação técnica e a facilidade para controlar esses elementos no código.

Mesmo sendo pessoas de áreas diferentes e com responsabilidades distintas, todos os envolvidos na construção ou manutenção de um produto — e que utilizarão o design system — devem consultar a documentação, seja programadores ou designers. Ninguém é obrigado a decorar tudo, e essa é justamente a grande vantagem da documentação: ela permanece disponível para que o time a consulte sempre que necessário.

Estrutura convencional

Assim como cada marca é única, cada design system também é. No entanto, ao analisarmos a documentação de alguns dos mais conhecidos — como Material Design, Atlassian e Carbon — percebemos uma grande semelhança na forma como as informações são organizadas. Normalmente, você encontrará as seguintes categorias:

1. ***Get started***: o ponto de partida para começar a utilizar o design system pela primeira vez, seja você uma pessoa designer ou desenvolvedora.
2. ***Foundation***: traz a documentação de todos os estilos visuais, acessibilidade e design tokens.
3. ***Components***: semelhante ao anterior, mas traz a documentação de uso de todos os *base components* do design system.

4. ***Patterns***: conta com padrões de uso de componentes para cenários ou páginas.

Nos capítulos anteriores, discutimos os elementos que compõem o *foundation* e os componentes. Cada um deles possui particularidades, e, nos próximos capítulos, aprofundaremos principalmente a documentação de componentes e *patterns*.

Como um design system será utilizado por profissionais de diferentes áreas, é essencial que todos saibam o que precisam para começar a utilizá-lo. Então, vamos ao ponto de partida.

Get started

Primeira vez com o design system

A área conhecida como *Get started* tem a proposta de responder a dúvidas sobre como começar a trabalhar nos produtos usando o design system. Não é uma página que as pessoas vão acessar com frequência. Na verdade, é provável que elas acessem somente uma vez, para conferir o tutorial de instalação. Por esse motivo, não inclua informações que não sejam essenciais. O foco da documentação estará nos outros tópicos. Concentre-se em ensinar designers e desenvolvedores a iniciarem o uso do design system.

Para ilustrar, trago o exemplo do Gestalt: na página "Designers", além de ensinar como conectar as diferentes bibliotecas no Figma, ele recomenda alguns plugins que podem ser utilizados para facilitar a produção de interfaces.

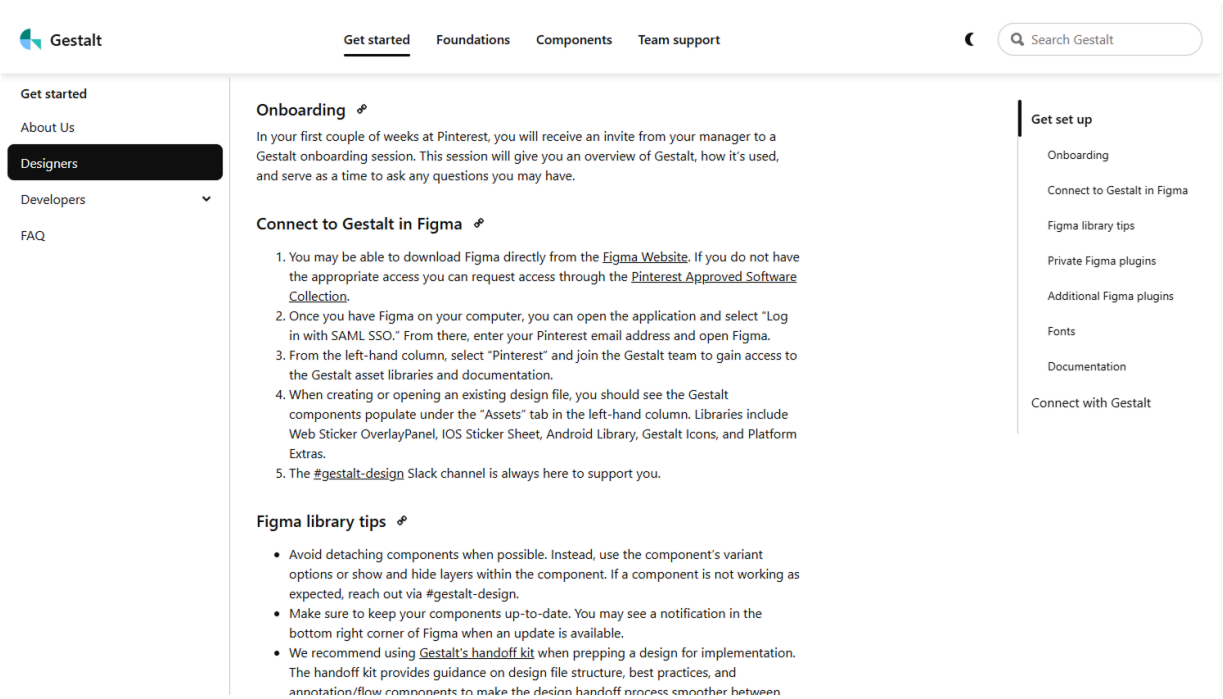


Figura 22.1: Página da seção 'Get started' da documentação do Gestalt, design system do Pinterest. O conteúdo está dividido em um menu lateral esquerdo com as opções 'Designers', 'Developers' e 'FAQ', e um painel principal com instruções de onboarding para designers, explicando como conectar o Figma ao Gestalt. Há também dicas sobre bibliotecas do Figma e links úteis como Slack e o site do Figma.

Você também pode encontrar esse tipo de informação em outras documentações em uma página chamada "*Tools*" (Ferramentas), que lista quais softwares, plugins, extensões e outros recursos devem ser instalados por cada profissional. Organize da forma que achar mais interessante para quem for acessar a documentação, mas não há problema nenhum em manter no *Get started*.

Transitando entre temas diferentes

Além da etapa que ensina como cada profissional fará para começar a trabalhar com o design system no seu contexto, há algumas outras informações que também podem ser inseridas nesse espaço e que valem ser mencionadas.

Hoje, é praticamente impossível encontrar um design system que não permita aplicações em temas diferentes — ou seja, modo claro e escuro. Contudo, além desses dois temas comuns, é normal que um design system tenha que se adaptar a diferentes marcas, mudando uma série de atributos visuais além das cores.

Se isso fizer parte do seu contexto, é importante incluir um tópico sobre como realizar essa transição, destacando principalmente as diferenças na abordagem de designers e desenvolvedores.

Migrando para uma nova versão

Conforme o design system é atualizado, em algum momento ele pode atingir um estágio de mudanças significativas, que não serão mais compatíveis com a versão atual. Trata-se de algo mais do que uma atualização comum: tokens podem ter sido renomeados, componentes completamente reestruturados e até mesmo APIs modificadas.

Ter uma seção para responder às questões sobre uma versão nova é importante, pois ajudará os colaboradores a entenderem quais foram as atualizações, como:

- Quais as novidades? Novos design tokens, componentes ou patterns;
- Quais são as melhorias? Por exemplo, um sistema de espaçamento fluido;
- O que foi removido ou substituído? Elementos obsoletos na biblioteca, por exemplo.

Além de informar sobre as mudanças, esta página deve guiar os colaboradores sobre como começar a trabalhar com a nova versão do design system, considerando o contexto de cada um. Isso pode incluir novas instalações, configurações do ambiente de desenvolvimento e a validação da consistência visual.

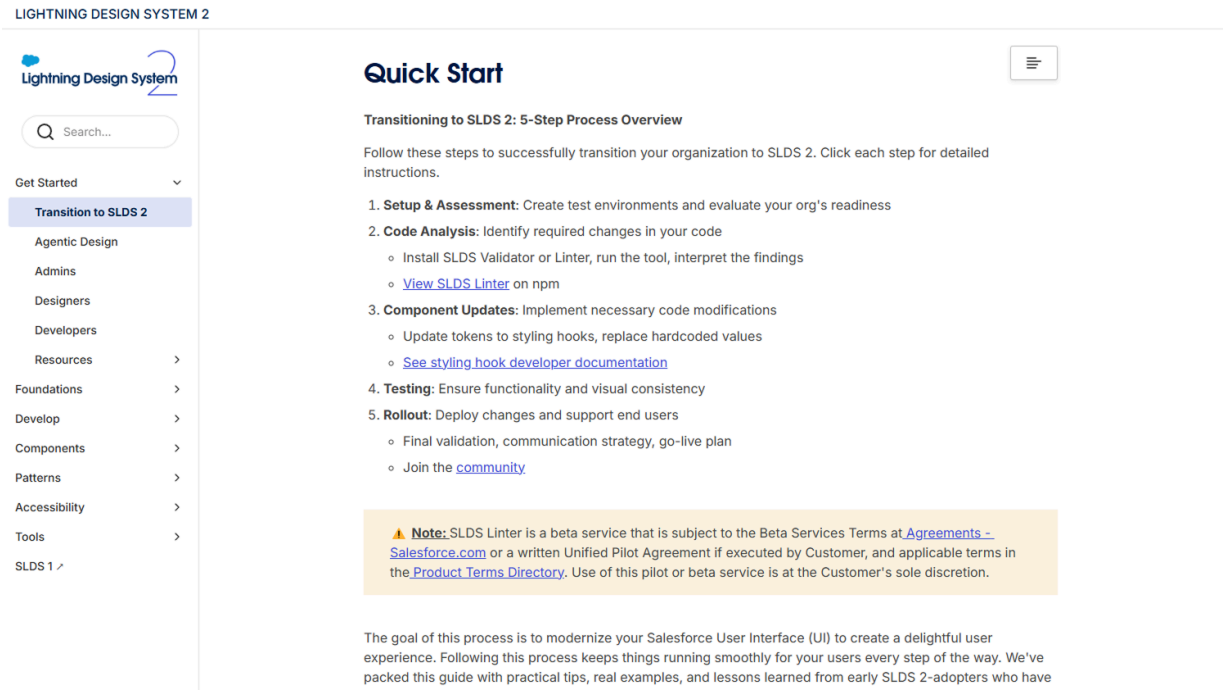


Figura 22.2: Captura de tela da página "Quick Start" do Lightning Design System 2 da Salesforce. O conteúdo descreve os 5 passos para transição para o SLDS 2: Setup & Assessment, Code Analysis, Component Updates, Testing e Rollout. Na lateral esquerda, há um menu de navegação com itens como "Transition to SLDS 2", "Foundations", "Develop", "Components", "Patterns" e outros. Na parte inferior do conteúdo principal, há um aviso em destaque sobre o SLDS Linter ser um serviço beta sujeito aos termos da Salesforce.

Contribuição

Mesmo que o seu cenário seja de uma “euquipe”, ou seja, uma equipe de uma pessoa só atuando no design system, é ótimo abrir a possibilidade para que mais pessoas possam contribuir com a evolução.

O tópico de contribuição permite que outras pessoas da empresa colaborem com melhorias de forma organizada. Ele pode estar aberto não apenas para designers e desenvolvedores, que são os principais envolvidos, mas também para outros profissionais de produto, desde que identifiquem um problema ou uma oportunidade que possa ser resolvida pelo design system.

As contribuições podem ocorrer em diferentes níveis: desde pequenos detalhes, como bugs, erros em um componente no Figma

ou a sugestão de um novo ícone, até melhorias mais significativas, como novos componentes ou funcionalidades.

Avalie a maturidade do seu design system, indique como os colaboradores podem contribuir e, se necessário, crie um *checklist* que garanta que as sugestões seguem as práticas estabelecidas.

CAPÍTULO 23
Documentando componentes

No seu design system, não importa se existem três estilos de botões diferentes, *cards* que podem ou não apresentar elevação e diversos campos de formulário, se não houver documentação adequada. Qualquer página que for trabalhada será repleta de componentes, sejam eles *core* ou *team components*.

Por mais que muitos desses componentes possam ser simples de utilizar, como é o caso dos botões, eles ainda podem apresentar formas de aplicação que só devem ser usadas com cuidado ou em determinados cenários. A documentação desses componentes é o que suprirá todas as dúvidas a respeito de como utilizá-los.

A forma como a documentação é escrita e organizada varia de design system para design system. Não existe um caminho perfeito para isso, mas um ponto interessante sobre a documentação de componentes é que, por mais que cada um seja diferente, eles possuem propriedades semelhantes. Isso abre a possibilidade de escrever a documentação de todos os componentes seguindo um roteiro padrão. É quase como contar a mesma história, apenas mudando o protagonista.

Considere que nada está escrito em pedra, mas, com base no conhecimento de design systems em que já trabalhei e até mesmo tomando como referência documentações de grandes design systems, você pode trabalhar com os seguintes tópicos:

1. Guia de uso.
2. Customizações.
3. Boas práticas.
4. Guia de conteúdo.

Note que essa sequência trata muito mais do aprendizado desse componente de forma geral. Defendo a ideia de que não são apenas os designers que devem saber utilizar esse componente, mesmo que a etapa de design de interface seja anterior à de desenvolvimento.

Entenda a seguir cada um dos quatro itens desse roteiro e como documentar seu componente da melhor forma.

Guia de uso

A primeira informação que as pessoas precisam ler sobre o seu componente é como utilizá-lo corretamente. Lembre-se de que essa é uma das seções mais importantes, portanto, sua descrição precisa ser completa, explicando para que serve o componente, em quais casos deve ser usado e sempre incluindo imagens.

Contudo, nem todos os componentes possuem apenas uma variante. Alertas, botões e vários outros componentes comuns possuem duas ou mais formas de serem utilizados, exigindo que a explicação de uso seja dividida pelas variantes.

Se for facilitar seu processo de documentação, para cada componente você pode responder às seguintes perguntas:

1. Qual é o propósito principal deste componente ou variante?
2. Em que situações ou contextos este componente ou variante deve ser usado?

As documentações dos componentes do Carbon Design System, na minha opinião, são referências em boas explicações. Mais do que apenas informar o que é o componente, elas indicam quando você deve ou não utilizá-lo. Para os casos em que o componente possui duas ou mais variantes, há uma seção adicional com imagens de cada variante e uma tabela explicando, de forma minuciosa, o caso de uso de cada uma delas.

Para incrementar ainda mais o seu guia de uso, seja o seu componente com uma ou mais variantes, você pode adicionar uma seção referente à **anatomia do componente**.

A anatomia tem como papel fundamental oferecer uma análise detalhada e aprofundada de cada parte que compõe o componente. Em vez de trazer uma visão superficial, cada elemento presente no componente é sinalizado de forma individual, para que os colaboradores saibam do que aquele componente é composto, para que serve e qual a nomenclatura correta.

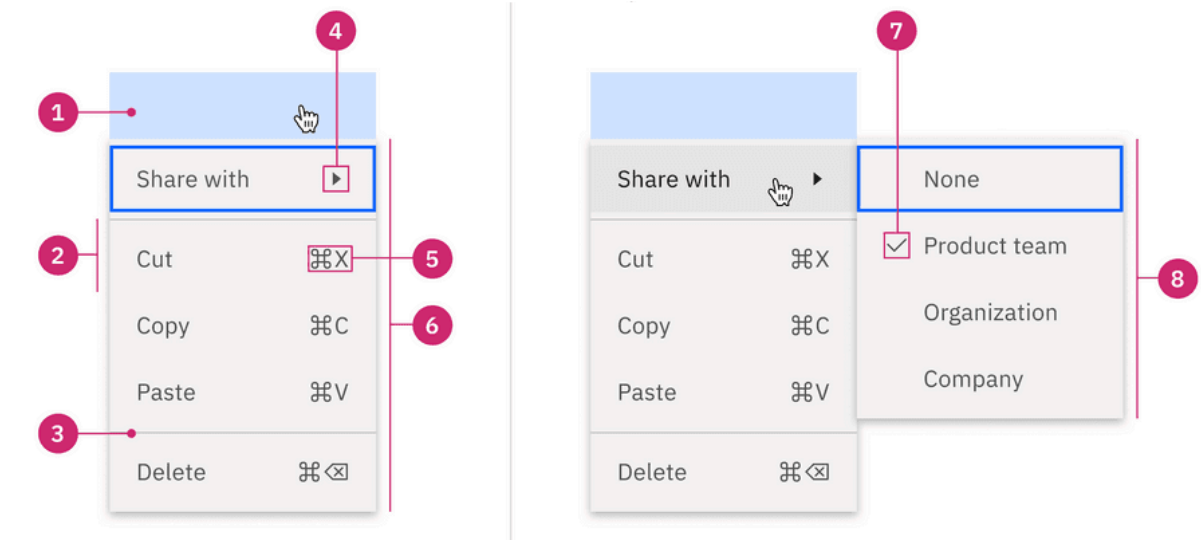


Figura 23.1: Anatomia de um componente de menu dropdown. O menu possui os itens "Compartilhar com", "Cortar", "Copiar", "Colar" e "Deletar". À esquerda, está uma versão do menu com o "Compartilhar com", sendo o único que possui submenus, mas está fechado. Há indicações numéricas ao redor que apontam para os locais do menu ajudando a entender os pontos anatômicos. À direita, está a versão do submenu de "Compartilhar com" aberto, mostrando as opções "Ninguém", "Time de produto", "Organização" e "Empresa"

Propriedades

Como você já leu no capítulo 21, um único componente pode conter inúmeras propriedades e formas de customização. Como a documentação já introduz o que é e para que serve o componente, esta seção tem o objetivo de especificar em quais cenários essas customizações podem ser utilizadas e bem aproveitadas.

Para facilitar o entendimento, organize todas as customizações que o seu componente pode ter por categorias, semelhante às citadas anteriormente — cor, tamanho e ícone — lembrando que, se necessário, você pode e deve incluir outras categorias.

Utilize sempre imagens para representar o componente ou, se viável, aplique o próprio componente diretamente na documentação. Quanto à descrição, evite que seja longa e escreva apenas o caso de uso daquela customização, incluindo, se possível, exemplos de aplicação nos produtos.

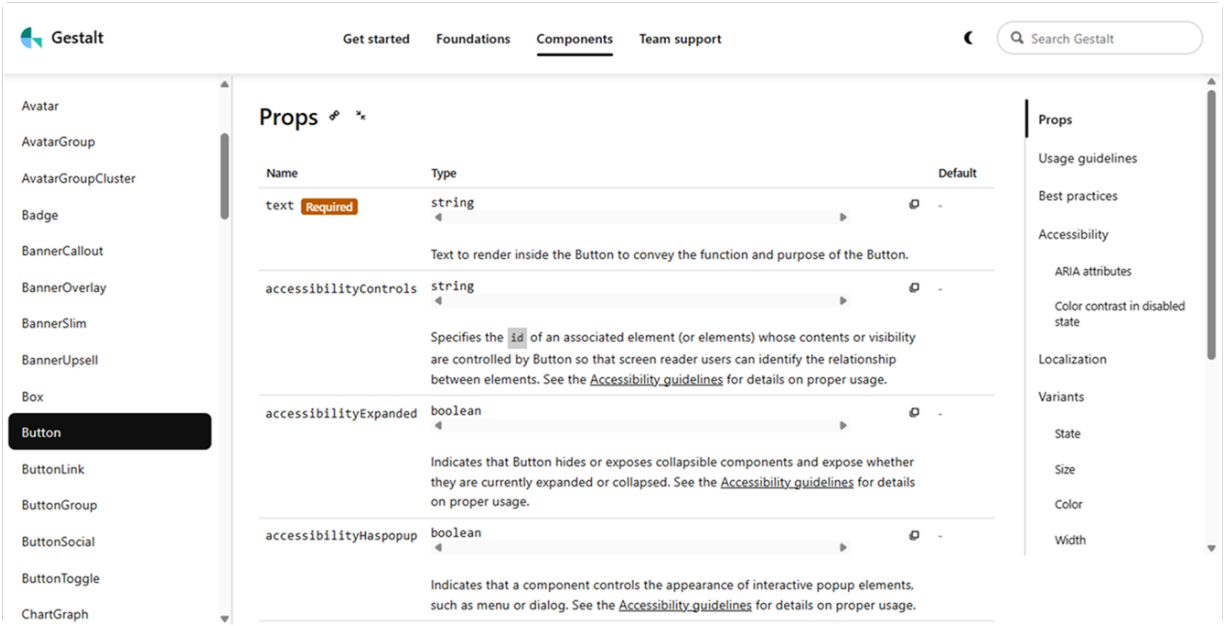


Figura 23.2: Página do componente Button do Gestalt Design System. No conteúdo da página está aparecendo o título "Props", além de 4 de 17 propriedades: text, acessibility controls, acessibility expanded e accessibility has popup.

Boas práticas

As boas práticas, também chamadas de "*dos and don'ts*" ou *patterns*, são a seção em que o time do design system indicará as formas adequadas de utilizar, os cuidados a serem tomados ou situações em que não devem ser usados de determinada maneira.

Os *"dos"* geralmente apresentam exemplos práticos e aplicáveis de como o componente deve ser usado corretamente. Isso inclui a aplicação adequada de propriedades, interações recomendadas e cenários ideais para a utilização do componente. Os exemplos positivos servem como guias tangíveis para os designers, fornecendo um entendimento claro das intenções de design.

Os *"cautions"* buscam mostrar aplicações dos componentes que podem ser utilizadas, mas em formato de alerta; ou seja, para que o designer tome cuidado ao usar o componente. Geralmente, envolvem casos de uso específicos ou contextos incomuns, que, embora possam ser usados em determinado cenário, nos demais podem não ser a melhor forma de aplicação.

Por outro lado, os *"don'ts"* apontam situações em que o componente pode ser mal utilizado ou cuja aplicação pode resultar em inconsistências. Isso pode incluir erros comuns, como dimensionamento inadequado, cores inapropriadas ou aplicação incorreta em certos contextos. Essas advertências são valiosas para designers e desenvolvedores, ajudando-os a evitar armadilhas comuns e a manter a integridade do produto.

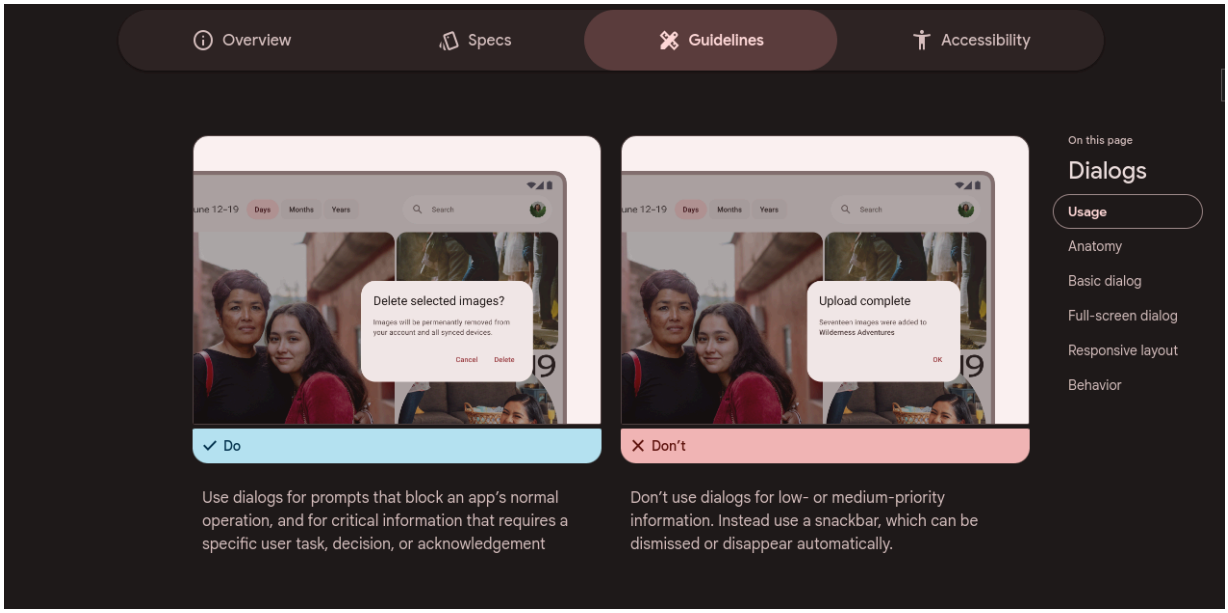


Figura 23.3: Página da documentação do componente Dialog do Material Design 3. Na página, é possível ver duas imagens lado a lado: a da esquerda está classificada como "do", indicando que os Dialogs devem ser usados para operações críticas ou informações importantes, enquanto a da direita é classificada como "don't", mostrando que não devem ser usados para informações de baixa ou média prioridade.

Por mais que todo componente do design system possua uma série de boas práticas e padrões, este caso pode se estender ainda mais e incluir cenários adicionais, mas falaremos melhor sobre isso no próximo capítulo.

Guia de conteúdo

Nem só de casos de uso, interações e customizações vive um componente. Todos os componentes possuem alguma forma de comunicação e quase todos incluem textos. Mesmo que o espaço para uma informação textual seja de uma ou duas palavras, é essencial que a mensagem no elemento seja definida corretamente, para que a pessoa usuária não tenha dúvidas sobre o que está ou estará fazendo. Para facilitar esse processo, documentamos um guia de conteúdo.

O guia de conteúdo é mais uma seção na documentação de componentes que tem por objetivo fornecer as diretrizes específicas sobre como criar, estruturar e apresentar o conteúdo da maneira correta.

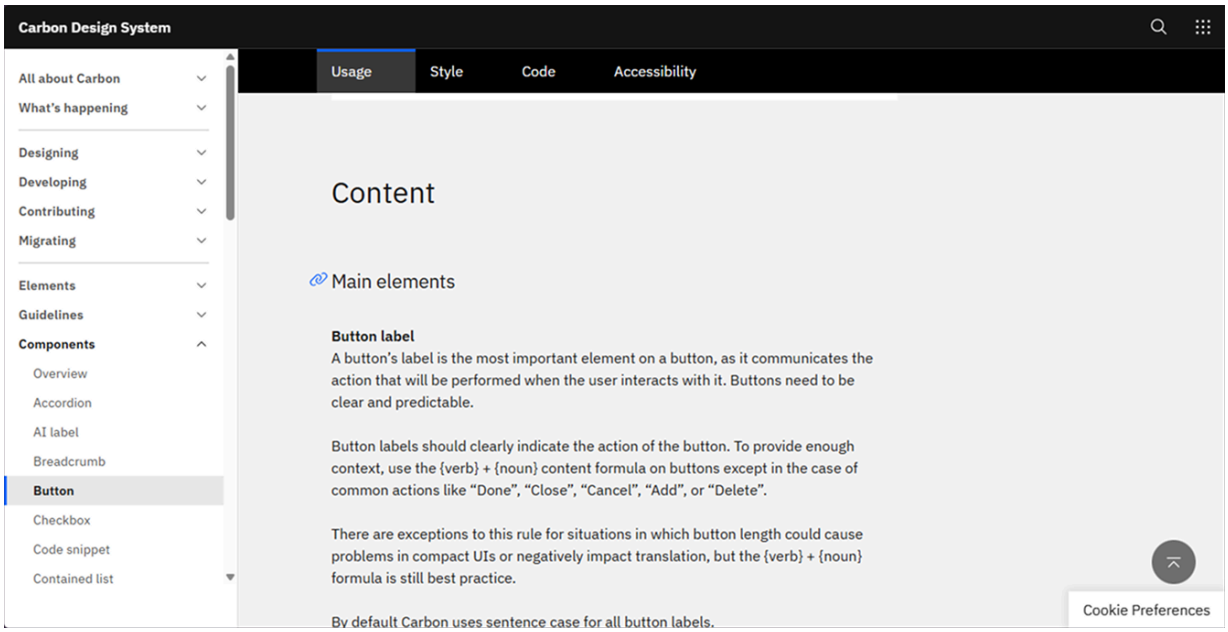


Figura 23.4: Página do componente Button do Carbon Design System, na seção Usage. No conteúdo da página está sendo exibido o guia de conteúdo com as instruções para o texto do botão.

Sua abordagem mais comum é relacionada à elaboração de mensagens textuais, mas um conteúdo pode ser apresentado de diferentes maneiras. Portanto, mídias como imagens e vídeos também devem fazer parte do guia se for necessário.

Para marcas que já possuem manuais de tom e voz, isso se torna uma "mão na roda" enorme na construção dos guias de conteúdo, pois, enquanto o manual estabelece os princípios e a personalidade verbal da marca, o guia de conteúdo traduz esses conceitos abstratos para a prática, fornecendo orientações ideais.

Se o tom da marca enfatiza a simplicidade e a empatia, o guia de conteúdo especificaria como esses princípios se manifestam em situações práticas. Ele alinha toda a equipe com o tom desejado,

garantindo que cada interação com o usuário reflita consistentemente a personalidade da marca.

Embora a maior frequência seja em elementos visuais, os guias de conteúdo também podem e devem contemplar a acessibilidade das informações para pessoas que utilizam softwares de leitura de tela.

Como imagens, ilustrações e ícones podem ser usados tanto em componentes do design system quanto ao longo das páginas de forma avulsa, torna-se importante que cada uma dessas mídias possua uma descrição detalhada, que, no desenvolvimento, é aplicada usando o atributo `alt`.

Para exemplificar uma aplicação real em um componente comum, vamos usar o *text input* (https://carbondesignsystem.com/components/text-input/usage/#content) do Carbon Design System. Nesse componente, a IBM fornece uma série de informações, como label, placeholder e texto de ajuda (referido como *helper text*). Para cada um desses itens, há direcionamento sobre como escrevê-los de forma ideal, incluindo o uso de letras maiúsculas, textos curtos e a recomendação de não usar dois pontos no caso específico da *label*.

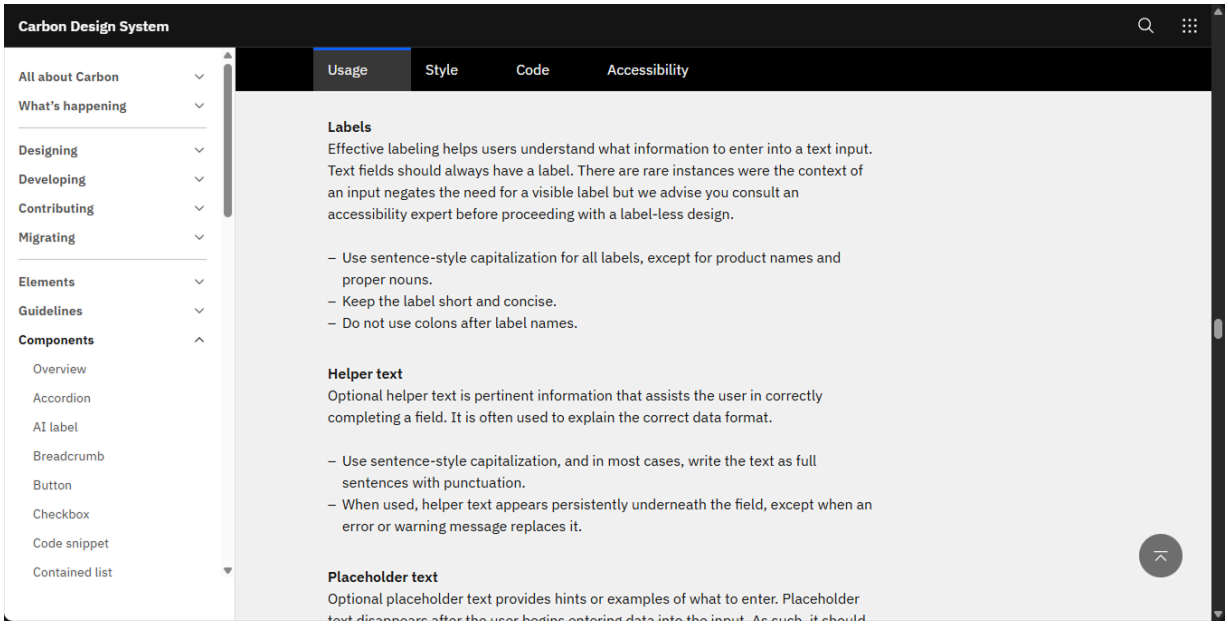


Figura 23.5: Página do componente Text input do Carbon Design System. Na página, é possível ver uma parte do guia de conteúdo indicando sobre como escrever para labels, textos de ajuda e placeholder.

Capítulo 24
Documentando os patterns

A tecnologia da web evoluiu ao ponto de termos uma série de sistemas disponíveis em nossas mãos. Isso ocorre independentemente de você usar Mac ou Windows como sistema operacional no computador, Android ou iOS no smartphone, ou até mesmo comprar na Amazon em vez do AliExpress. Por mais que as interfaces visuais sejam completamente diferentes, ainda existe uma similaridade entre elas.

Quando o assunto é comportamento, o ideal é trabalharmos com soluções chamadas de *patterns*, com as quais a pessoa usuária já saiba interagir, em vez de reinventar a roda com algo que obrigue uma pessoa a pensar no que ela deve fazer, complicando ainda mais a usabilidade em vez de facilitá-la.

Os *patterns*, ou padrões (em português), são soluções reutilizáveis e eficazes para problemas ou cenários comuns na interface do usuário (UI), como registro de conta, filtros de busca, *wizards* e até mesmo carrinhos de compra.

Todos esses exemplos, assim como muitos outros, são situações cotidianas que já executamos inúmeras vezes e que, quando paramos para refletir, mostram que raramente encontramos algo mirabolante que fuja do padrão já conhecido.

No contexto de um design system, os *patterns* podem agir em ocasiões diferentes. Podem ser aplicados tanto a fluxos de trabalho, que unem uma série de elementos em um determinado contexto, quanto a interfaces únicas, como componentes, que também possuem seus próprios padrões.

O design system da Wise tem uma ótima coleção de *patterns* que variam nessas duas situações, tendo guias sobre como trabalhar

corretamente com o componente de notificação (https://wise.design/patterns/notifications) e sobre como escrever mensagens de validação (https://wise.design/patterns/validation-messages) corretamente para os campos de formulário ou até mesmo para artigos de ajuda (https://wise.design/patterns/help-articles).

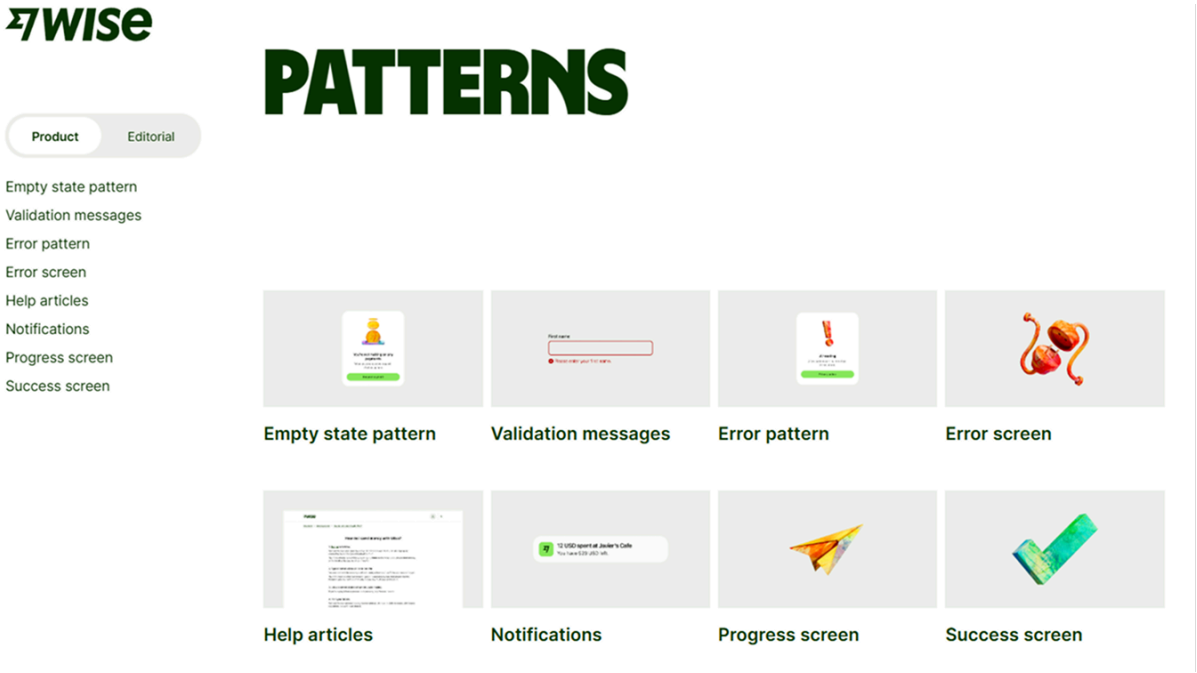


Figura 24.1: Página de patterns do Wise Design. No centro da página, estão os cards: padrões de empty state, mensagens de validação, padrões de erro, páginas de erro, artigos de ajuda, notificações, página de progresso e página de sucesso.

Você pode se guiar por padrões populares e frequentemente utilizados em ambientes digitais. Como referência, recomendo o site *UI-Patterns.com* (https://ui-patterns.com). No entanto, é importante perceber que cada marca possui sua própria forma de produzir, que pode diferir das demais, e, como já mencionado, deve sempre manter comportamentos que sejam familiares ao usuário.

24.1 Definindo padrões

Se você está nas fases iniciais de um design system, atuando em sua construção antes do lançamento, é altamente recomendável que você (e sua equipe, se houver) introduza *patterns*. Isso ajudará a evitar que diferentes produtos que utilizem o design system, embora visualmente semelhantes, acabem com componentes ou cenários teoricamente idênticos aplicados de maneiras diferentes.

Caso você esteja trabalhando em um design system já lançado, mas que ainda não possui *patterns* definidos ou documentados, recomendo começar pelos componentes. Essas definições são fundamentais para estabelecer padrões consistentes nos fluxos de trabalho.

Novos *patterns* não devem surgir do além e muito menos de achismos, mas sim da identificação de problemas recorrentes entre as aplicações. Por exemplo, se diferentes formulários possuem campos obrigatórios, mas cada um apresenta mensagens de validação diferentes quando um campo não é preenchido, deve-se estabelecer uma mensagem padrão.

A estrutura de uma página ou seção de um *pattern* pode variar de uma documentação para outra, e não existe um roteiro ideal para sua descrição. No entanto, ela apresenta similaridades com a forma como um componente é documentado.

Tradicionalmente, *patterns* precisam de um nome e de uma descrição que expliquem qual problema eles resolvem e como o fazem. Referencie todos os componentes e elementos que possam fazer parte desse padrão e, para cada aplicação possível, inclua um exemplo. Imagens que mostrem aplicações reais do *pattern* tornam seu entendimento e uso mais fácil para os designers.

Um artigo que recomendo para ampliar este tópico sobre a documentação de *patterns* é *How to define and document patterns in your design system* (https://zeroheight.com/help/guides/how-to-define-and-document-patterns-in-your-design-system/) — em

português: "Como definir e documentar padrões em seu sistema de design" —, do Zeroheight, escrito por Michelle Chin. Nele, são citadas diversas informações que podem ser incluídas nessa documentação, tais como:

- **Diretrizes de uso**: explica por que e quando o *pattern* deve ser utilizado, incluindo boas práticas e o que evitar para garantir que ele seja aplicado corretamente.
- **Diretrizes de escrita**: define o tom e a voz das mensagens relacionadas ao *pattern*, além de apresentar padrões de linguagem, como formatos de mensagens de erro ou instruções.
- **Comportamentos de interação**: descreve como o usuário interage com o *pattern*, incluindo fluxos, animações ou reações visuais e o que acontece em cada etapa dessa interação.
- **Exemplos na prática**: mostra capturas de tela ou links de interfaces reais onde o *pattern* é usado, permitindo que as pessoas vejam como ele se comporta em diferentes contextos.
- **Acessibilidade**: aponta como o *pattern* lida com navegação por teclado, leitores de tela, ordem de leitura e outros requisitos para garantir que ele seja acessível a todas as pessoas.
- **Localização e responsividade**: define como o *pattern* se adapta a diferentes tamanhos de tela e idiomas, considerando variações de conteúdo e de layout.
- **Cenários extremos**: documenta casos fora do comum que desafiam o uso padrão, justificando quando e por que fugir da regra pode ser necessário e como lidar com isso.
- **Referência à pesquisa de usuário**: adiciona links ou dados de pesquisas que embasaram o *pattern*, ajudando a contextualizar sua criação com base em testes ou feedbacks reais.

- **Informações específicas do desenvolvedor**: reúne instruções técnicas para desenvolvedores implementarem o *pattern* corretamente, como requisitos de código, dependências ou limitações.
- **Contexto histórico das decisões**: traz o histórico de decisões que levaram à criação do *pattern*, explicando problemas enfrentados e por que essa solução foi escolhida, sendo útil para manter a intenção original em futuras revisões.

Para fechar, é fundamental que você e todos que trabalham com um design system entendam que *patterns*, assim como os demais elementos abordados ao longo dos capítulos, também evoluem. Vivemos em uma era de rápido avanço tecnológico, em que as expectativas dos usuários estão em constante crescimento. Como responsável pelo design system, é seu dever revisitar periodicamente os *patterns*, garantindo que continuem bem adaptáveis e relevantes.

Parte 7 - Conclusão

Design system não é um assunto novo no mundo, mas ainda é muito recente no Brasil, e não só há muito para se aprender, como também muito para continuar evoluindo.

Uma das coisas mais importantes que quero que você leve para o seu dia a dia profissional relacionado ao tema do livro é que design system não é apenas um mero projeto paralelo, que ficará pronto em duas semanas a um mês. Estamos falando de um produto robusto, complexo e escalável, e que muitas vezes exigirá uma equipe dedicada para desenvolvê-lo.

A maneira como escrevi e organizei os capítulos não foi pensada, necessariamente, para uma leitura contínua. Claro, nada impede esse formato de leitura, mas eu imagino esse livro mais como um guia, com frases sublinhadas e post-its colados, para consultar quando houver necessidade.

Como muitos livros e escritores que me inspiraram na escrita deste, não imagino que esta seja a primeira nem a última vez que publico sobre design system. De todas as áreas e assuntos relacionados a design, este é o que continua me dando mais vontade de explorar e estudar. Portanto, espere futuras edições, pois elas virão!

E chegamos ao fim deste livro. De todo coração, espero que a leitura tenha sido tão gratificante para você quanto a escrita foi para mim. Se quiser continuar aprofundando o assunto deste livro, meu chat está mais do que aberto para conversas e trocas de ideias. Fique à vontade para me chamar pelo LinkedIn ou, se preferir, minhas outras redes sociais estão no meu site:

- https://mateusvillain.com/
- https://www.linkedin.com/in/mateusvillain/

Capítulo 25

Glossário

A

- **Atomic Design** — Metodologia criada por Brad Frost que organiza a criação de interfaces em níveis: átomos, moléculas, organismos, templates e páginas.
- **Auditoria** — Revisão periódica para avaliar consistência, usabilidade e aderência do design system nos produtos.

B

- **Brand Guidelines** — Documentação que define cores, tipografia, tom de voz e identidade visual.
- **Breakpoints** — Pontos definidos no design responsivo para alterar o layout e garantir adaptação em diferentes telas.

C

- **camelCase** — Convenção de nomenclatura em que a primeira palavra é minúscula e as seguintes iniciam com maiúscula, sem espaços (ex.: `primaryButton`).
- **Componente** — Elemento reutilizável da interface (botão, card, modal) que segue regras de design e código.
- **Component Library** — Conjunto de componentes documentados e prontos para uso em produtos.
- **Consistência** — Característica de manter o mesmo padrão em diferentes interfaces e produtos.
- **Cross-platform** — Capacidade de um design system funcionar em múltiplas plataformas (web, iOS, Android).

D

- **Dark Mode** — Tema escuro aplicado como variação de estilo.
- **Design Inclusivo** — Prática que considera diversidade de usuários no processo de design.
- **Design Language** — Linguagem visual que une princípios de design, estilo e identidade da marca.

- **DesignOps** — Prática focada em melhorar processos, ferramentas e colaboração no design.
- **Design Principles** — Regras norteadoras que orientam decisões visuais e de experiência.
- **Design Responsivo** — Capacidade de adaptar interfaces para diferentes tamanhos de tela.
- **Design Token** — Variáveis que armazenam valores de estilo (cores, espaçamento, tipografia, bordas), garantindo consistência e fácil atualização.
- **Developer Experience (DX)** — Experiência dos desenvolvedores ao consumir o design system.
- **Documentação** — Guia que explica como usar, manter e contribuir com o design system.

E

- **Ecossistema** — Conjunto de produtos, ferramentas e fluxos que utilizam o design system.
- **Elevation** — Camada visual usada para representar profundidade, como sombras.
- **em** — Unidade relativa de medida no CSS que se baseia no tamanho da fonte do elemento pai, permitindo ajuste proporcional do layout e tipografia.
- **Escala tipográfica** — Hierarquia definida de tamanhos de fontes para títulos, subtítulos e textos.
- **Estados de interação** — Diferentes estados visuais e comportamentais de um componente (ex.: hover, active, disabled, focus).

F

- **Figma** — Ferramenta de design colaborativa baseada na nuvem, muito usada para criar interfaces, protótipos e bibliotecas de componentes.
- **Figma Library** — Biblioteca compartilhada de componentes dentro do Figma.

- **Fonte** — Conjunto de caracteres (letras, números, símbolos) com um estilo específico usado em interfaces e documentos.
- **Foundation** — Camada base do design system: cores, tipografia, grid, ícones, espaçamento.
- **Framework** — Estrutura de código que serve de base para criação de produtos (React, Vue, Angular) e onde o design system pode ser implementado.

G

- **Gap** — Espaçamento entre elementos dentro de um container, como colunas ou linhas, usado principalmente em layouts de grid e flexbox.
- **Governança** — Estrutura de decisões, papéis e responsabilidades na manutenção do design system.
- **Grid** — Estrutura de colunas e espaçamentos que organizam layouts de forma consistente.
- **Guidelines** — Regras e boas práticas para usar corretamente componentes e padrões.

H

- **Hand-off** — Processo de transição do design para desenvolvimento, onde o design system facilita a comunicação.
- **Hierarquia Visual** — Organização que define o peso e a importância de elementos visuais.
- **Hotfix** — Correção rápida aplicada em uma versão do design system.

I

- **Iconografia** — Conjunto padronizado de ícones usado em produtos.
- **Identidade Visual** — Conjunto de elementos que representam a marca (cores, fontes, ícones, logotipos).

K

- **Kebab Case** — Convenção de nomenclatura em que palavras são separadas por hífens (ex.: `primary-button`).

L

- **Layout Grid** — Sistema de colunas usado para estruturar layouts responsivos.
- **Library** — Coleção organizada de recursos: tokens, ícones, componentes, templates.
- **Light Mode** — Tema claro aplicado como variação de estilo, geralmente usado em oposição ao Dark Mode.

M

- **Maintenance** — Práticas para manter o design system atualizado e escalável.
- **Margem** — Espaço externo de um elemento, separando-o de outros elementos ao seu redor.
- **Modularidade** — Estrutura que permite usar partes independentes do design system sem depender de todo o conjunto.
- **Motion Guidelines** — Regras para uso de animações e microinterações consistentes.
- **Multi-brand** — Prática de um design system suportar múltiplas marcas, permitindo variações visuais e de identidade dentro da mesma base de componentes.

N

- **Naming Convention** — Padrão de nomenclatura para tokens, componentes e variáveis.
- **Neutral Colors** — Cores base como cinzas, usadas em fundos, textos e bordas.

O

- **Open Source** — Quando o design system é disponibilizado publicamente para uso e colaboração externa.

P

- **Padding** — Espaço interno de um elemento, entre o conteúdo (texto, ícone) e a borda do próprio elemento.
- **Paleta de Cores** — Conjunto de cores primárias, secundárias e de suporte usado no design system.
- **Pascal Case** — Convenção de nomenclatura em que todas as palavras iniciam com maiúscula, sem espaços (ex.: `PrimaryButton`).
- **Pattern** — Solução recorrente para um problema de interface, como formulários, navegação ou autenticação.
- **Pixel** — Unidade de medida fixa usada em telas digitais para definir tamanhos de elementos, espaçamentos ou resolução de imagens.
- **Playground** — Ambiente interativo para testar e experimentar componentes.
- **Princípios** — Valores que orientam decisões de design e uso do sistema.

R

- **Release Notes** — Histórico de versões e atualizações do design system.
- **rem** — Unidade relativa de medida no CSS que se baseia no tamanho da fonte raiz (`html`), útil para escalabilidade e acessibilidade.
- **Repositório** — Local onde o código e documentação do design system são versionados.
- **Roadmap** — Planejamento das próximas entregas e evoluções do design system.

S

- **Semantic Version (SemVer)** — Sistema de versionamento que utiliza três números: Major.Minor.Patch (ex.: `1.2.3`) para indicar mudanças e compatibilidade.
- **Snake Case** — Convenção de nomenclatura em que palavras são separadas por underscore/underline (ex.: `primary_button`).
- **Spacing System** — Escala padronizada de espaçamentos usada em interfaces.
- **Storybook** — Ferramenta para documentar e testar componentes de forma isolada.
- **Style Guide** — Documento que centraliza diretrizes visuais e de interação.

T

- **Tipografia** — Conjunto de regras e escolhas de fontes, tamanhos, pesos e espaçamentos que definem a hierarquia visual do texto.
- **Theming** — Aplicação de diferentes estilos (claro/escuro, marcas distintas) sobre a mesma base de componentes.
- **Token Alias** — Token que referencia outro token, permitindo maior flexibilidade.

U

- **UI Kit** — Conjunto de elementos de interface em formato de design (ex.: Figma, Sketch).
- **Usage Guidelines** — Orientações de uso correto de componentes e padrões.

V

- **Variants** — Diferentes versões de um mesmo componente (ex.: botão primário, secundário, desabilitado).
- **Versionamento** — Gestão de diferentes versões do design system e seus componentes.
- **Viewport** — Área visível de uma página em diferentes dispositivos, usada no design responsivo.

W

- **Web Components** — Padrão de desenvolvimento que permite criar componentes reutilizáveis nativos da web.
- **White Label** — Adaptação do design system para ser usado por diferentes marcas, com personalização mínima.

Z

- **Zeroheight** — Ferramenta bastante usada para documentação de design system. Usada para criar e manter documentação de forma visual e colaborativa.

Capítulo 26
Referências

Referências de sucesso

Qualquer coisa no mundo só evolui quando diversas pessoas diferentes compartilham sobre o assunto.

A seguir, apresento uma lista de pessoas e comunidades que acompanho e que são referências em design systems e temas relacionados.

Pessoas

- Ana Takahashi: especialista em design systems e criadora do curso Design Tokens Advanced. (https://www.linkedin.com/in/anapaulatakahashi/)
- Brad Frost: autor do livro Atomic Design e um dos maiores nomes quando o assunto é design system. (https://bradfrost.com)
- Bruno Biagioni: Design System & Ops Lead e criador da Iniciativa UI. (https://www.linkedin.com/in/brunobiagioni/)
- Dan Mall: especialista, professor e consultor em design systems. (https://danmall.com/)
- Felipe Nzongo: designer e autor do primeiro livro em português sobre design systems, o Design em Escala. (https://www.linkedin.com/in/filipenzongo/)
- Glauber Laender: fundador da Escola Escala e do Detach! (https://www.linkedin.com/in/glauberlaender/)
- Guilherme Gonzales: designer especialista em design systems e DesignOps. (https://guigonzalez.com.br/)
- Liliane Claudia: designer e especialista em acessibilidade. (https://br.linkedin.com/in/lilianeclaudia)

- Lukas Oppermann: Product Designer e um dos responsáveis pelo Primer, design system do GitHub. (https://lukasoppermann.com)
- Marcelo Sales: especialista em acessibilidade digital e design inclusivo. Fundador da Tudo é Acessibilidade. (https://www.linkedin.com/in/msales78/)
- Nate Baldwin: especialista em Design Systems e um dos responsáveis pelo Spectrum, design system da Adobe. (https://natebaldw.in/)
- Nathan Curtis: especialista em design systems e uma das maiores referências na área. Fundador da EightShapes. (https://medium.com/@nathanacurtis)
- Rafael Coronel: cofundador do DesignBoost. (https://www.linkedin.com/in/orafaelcoronel/)
- Silvia Bormüller: responsável pelo Into Design Systems, conferência focada somente em design systems. (https://www.linkedin.com/in/silbormueller/)

Blogs e newsletters

- DesignSystems.com: https://www.designsystems.com
- Design Tokens Pills: https://designtokens.substack.com
- Design System University: https://designsystem.university/

Eventos

- Detach!: o primeiro — e, até o momento, único — evento 100% brasileiro sobre design systems. (https://www.detach.com.br/)
- Config : conferência anual da Figma, que em toda edição traz tópicos focados em design system. (https://config.figma.com/)
- Into Design Systems: uma das maiores e mais famosas conferências sobre design system no mundo. (https://www.intodesignsystems.com/)

Ferramentas e processos mencionados

Mencionei diversas ferramentas ao longo dos capítulos — muitas delas são plataformas online, outras são documentos — e seria um processo muito entediante ter que caçar cada uma delas. A seguir, listo todas as ferramentas que citei:

- Atmos: plataforma que gera paletas de cores utilizando o sistema de cor LCH. (https://atmos.style/color-picker/lch)
- Atomic Design: metodologia criada por Brad Frost. (https://atomicdesign.bradfrost.com/chapter-2/)
- ColorBox: ferramenta que cria paletas de cores utilizando conceitos de curvas de Bézier. (https://colorbox.io)
- Figma: a principal ferramenta de UI usada para criar e manter design systems. (https://www.figma.com/)
- GO Product Roadmap: método de roadmap orientado a objetivo, desenvolvido por Roman Pichler. (https://www.romanpichler.com/tools/the-go-product-roadmap/)
- Modular Scale: ferramenta para gerar escalas tipográficas usando o conceito de escalas modulares. (https://www.modularscale.com)
- Picking Parts, Products & People: dinâmica criada por Nathan Curtis para a priorização de um design system. (https://drive.google.com/file/d/1qXMUXKHaEXnLDOu99GCzTMY2XW6NnPe_/view)

Bibliografia

Neste livro, trouxe tanto vivências e opiniões pessoais sobre design systems quanto percepções e visões de outras pessoas — tanto daquelas que atuam diretamente na área quanto de outras que têm relação com o tema. Independentemente do caso, considero todas elas referências nesse cenário e recomendo as leituras a seguir para quem deseja se aprofundar ainda mais.

ADOBE. **Design Tokens**. Disponível em: https://spectrum.adobe.com/page/design-tokens/. Acesso em: 01 dez. 2023.

BABICH, Nick. **Graphical User Interface as a Reflection of the Real World: Shadows and Elevation**. Disponível em: https://uxplanet.org/graphical-user-interface-as-a-reflection-of-the-real-world-shadows-and-elevation-f456530317f4. Acesso em: 30 nov. 2023.

BABICH, Nick. **How To Use Shadows And Blur Effects In Modern UI Design**. Disponível em: https://www.smashingmagazine.com/2017/02/shadows-blur-effects-user-interface-design/. Acesso em: 12 mai. 2025.

BRINGHURST, Robert. **The Elements of Typographic Style**. 3. ed. Point Roberts, WA: Hartley & Marks, 2004.

CHIN, Michelle. How to define and document patterns in your design system. Zeroheight, 2024. Disponível em: https://zeroheight.com/help/guides/how-to-define-and-document-patterns-in-your-design-system/. Acesso em: 7 mai. 2025.

CURTIS, Nathan. **Naming Tokens in design systems**. Disponível em:

https://medium.com/eightshapes-llc/naming-tokens-in-design-systems-9e86c7444676. Acesso em: 01 dez. 2023.

CURTIS, Nathan. **Picking Parts, Products & People**. 22 mar. 2016. Disponível em: https://medium.com/eightshapes-llc/picking-parts-products-people-a06721e81742. Acesso em: 4 set. 2023.

DUBERG, Oskar. **Design Principles: How To Create & Use Them**. Disponível em: https://www.frontify.com/en/blog/design-principles-creation-and-usage/. Acesso em: 02 abr. 2024.

FILHO, Carlos Henrique. **Organizando componentes de time em um Design System**. Disponível em: https://medium.com/triadeecoo/organizando-componentes-de-time-em-um-design-system-f76d59cb5fe0. Acesso em: 12 jun. 2024.

GOOGLE. **Elevation**. Disponível em: https://m3.material.io/styles/elevation/overview. Acesso em: 29 nov. 2023.

GOOGLE. **Elevation**. Disponível em:

https://m2.material.io/design/environment/elevation.html. Acesso em: 12 mai. 2025.

HALL, David. **Design Detail: Crafting Better Shadows for Interaction**. Disponível em: https://uxdesign.cc/design-detail-crafting-better-shadows-for-interaction-b94796a29664. Acesso em: 30 nov. 2023.

JAMESON, John D. **Guidelines for Letterspacing Type**. Disponível em: https://johndjameson.com/posts/guidelines-for-letterspacing-type. Acesso em: 20 nov. 2023.

KALEY, Anna. **Content Inventory and Auditing 101**. 27 set. 2020. Disponível em: https://www.nngroup.com/articles/content-audits/. Acesso em: 4 set. 2023.

MOLES, Tony. **Why Is Modular Scale a Good Idea in Typography?**. Disponível em: https://digital.com/how-to-create-a-website/how-to-make-your-website-accessible-for-keyboard-only-visitors/. Acesso em: 26 ago. 2023.

PICHLER, Roman. **GO Product Roadmap**. Disponível em:

https://www.romanpichler.com/tools/the-go-product-roadmap. Acesso em: 4 set. 2023.

PM3. **Matriz RACI: o que é, benefícios e como elaborar**. Disponível em: https://www.cursospm3.com.br/blog/matriz-raci/. Acesso em: 05 mai. 2024.

STOCKS, Elliot Jay. **Choosing a suitable line height**. Disponível em: https://fonts.google.com/knowledge/using_type/choosing_a_suitable_line_height. Acesso em: 16 nov. 2023.

TOXBOE, Anders. **UI Patterns**. Disponível em: https://ui-patterns.com. Acesso em: 24 jan. 2024.

UXCEL. **UI Components & Patterns: Component States**. Disponível em: https://app.uxcel.com/courses/ui-components-n-patterns/component-states-499. Acesso em: 14 jan. 2024.

VIDRA, Vojtěch. PEŠIČKA, Ondřej. **LCH is the best color space!**. Disponível em: https://atmos.style/blog/lch-color-space. Acesso em: 11 jun. 2024.

ZHANG, Wenting. **Why Is Modular Scale a Good Idea in Typography?**. Disponível em: https://medium.com/odd-questions/why-is-modular-scale-a-good-idea-in-typography-a4eae0ef4a04. Acesso em: 21 out. 2023.